Development of a Software Testing Best Practice Framework for Medical Device Software

Andrzej Beniamin BUJOK

A thesis submitted in fulfilment of the
requirements for the degree of
Master of Science
to the

Dundalk Institute of Technology
School of Informatics and Creative Arts

Supervised by:

Dr Peadar GRANT
Dr Silvana Togneri MACMAHON
Prof Fergal MCCAFFERY

June 2020

Declaration

We, the undersigned declare that this thesis entitled *Development of a Software Testing Best Practice Framework for Medical Device Software* is entirely the author's own work and has not been taken from the work of others, except as cited and acknowledged within the text.

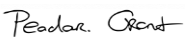
The thesis has been prepared according to the regulations of Dundalk Institute of Technology and has not been submitted in whole or in part for an award in this or any other institution.

Author Name: Andrzej Beniamin Bujok

Author Signature: 

Date: 05/05/2021

Supervisor Name: Peadar Grant

Supervisor Signature: 

Date: 05/05/2021

Acknowledgements

I would like to begin with giving my thanks and dedicating my thesis to my deceased parents for instilling in me the love of knowledge and art. They motivated me to educate myself at every stage of my life. I would also like to thank my beloved wife Dorota that she has never complained and always supported me during the challenging time of my study.

I am grateful to my supervisors for accompanying me on my research journey. First of all, I would like to thank Dr Silvana Togneri MacMahon, who has been supervising me since I made the first research step, through all subsequent research stages, up to the finish line of this study. Silvana, without your great contribution this study could not be completed. I also thank Dr Peadar Grant for his contribution, especially in terms of reviews and comments for my thesis. I thank Prof Fergal McCaffery that he gave me the opportunity to enter the world of science and commence research study. I appreciate my supervisors for leading me to continually develop research skills and knowledge. This journey was exciting when I experienced the taste of progress, but was challenging when I struggled with failures, and both, ups and downs affected my personal development.

I thank my colleagues, namely Hamsini, but also Surafel and others, for supporting me and each other in various aspects of our research. I also thank medical device software development organisation and software testing organisation for taking part in this research, especially in terms of validation. Finally, I would like to thank LERO, the Irish Software Engineering Research Centre, for supporting my research by providing funds for my studies.

Contents

List of Tables

List of Figures

List of Publications

Journals

Bujok, A.B., MacMahon, S.T., Grant, P., Whelan, D., Rickard, W.J. and McCaffery, F., 2017. Approach to the development of a Unified Framework for Safety Critical Software Development. *Computer Standards & Interfaces*, *54*, pp. 152-161. Available from: https://www.sciencedirect.com/science/article/abs/pii/S0920548916301921

Conference and Workshop Proceedings

Bujok, A.B., MacMahon, S.T., McCaffery, F., Whelan, D., Mulcahy, B. and Rickard, W.J., 2016, June. Safety critical software development–extending quality management system practices to achieve compliance with regulatory requirements. In *International Conference on Software Process Improvement and Capability Determination* (pp. 17-30). Springer International Publishing. Available from: http://eprints.dkit.ie/545/

Bujok, A.B., MacMahon, S.T., Grant, P., McCaffery, F., 2017, October. Approach to the Development of a Medical Device Software Quality Assurance Framework. In *11th Systems Testing and Validation Workshop (STV17) and 3rd International Workshop on User Interface Test Automation (INTUITEST 2017)* (pp. 9-23). Proceedings of Joint Research Workshop. Available from: http://publica.fraunhofer.de/documents/N-473344.html

List of Acronyms

DSR	Design Science Research
EU	European Union
FDA	Food and Drug Administration
IEC	International Electro-technical Commission
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organisation for Standardization
MED-V-STEP	Medical Device-Verification-Software Test Processes
MED-SQA	Medical Device-Software Quality Assurance
QA	Quality Assurance
RO	Research Objective
RQ	Research Question
RSQ	Research Sub-Question
SDLC	Software Development Life-Cycle
SLCPs	Software Life-Cycle Processes
SQA	Software Quality Assurance
U.S.	United States of America
V&V	Verification & Validation

Development of a Software Testing Best Practice Framework
for Medical Device Software
Andrzej Beniamin Bujok

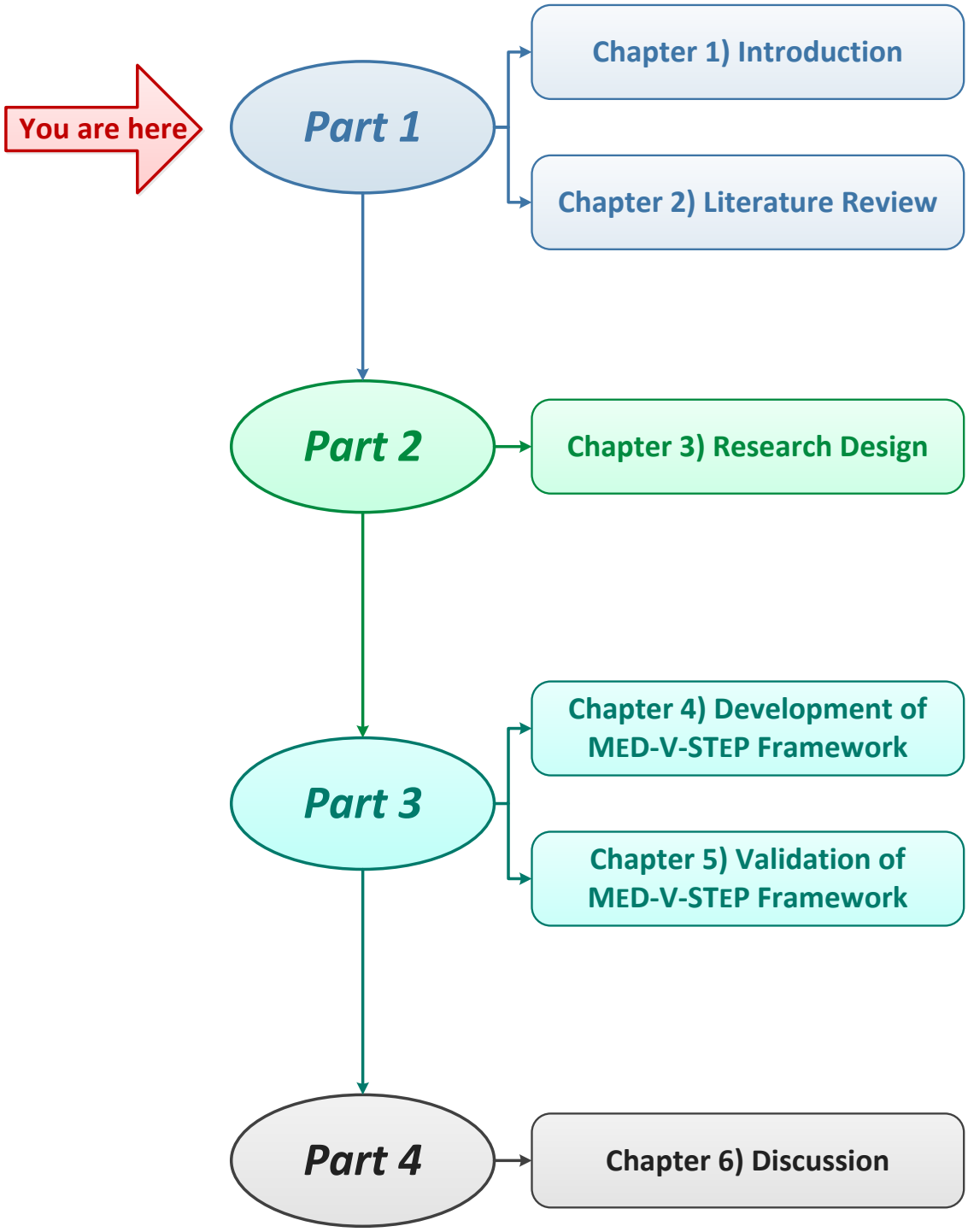
Abstract

Software testing is essential to maintain a high quality of software. This is particularly true in the medical device domain where software quality is closely related to safety, and software failure can cause injury or death to a patient. The increasing number of adverse events and recalls of medical devices due to software failures demonstrates the need to improve the safety of medical device software. Organisations that develop medical device software are required to test increasingly complex software and detect an increasing number of defects. Software safety can be improved by advancing the testing efficiency in detecting software defects that could result in medical device malfunction.

International standardisation organisations address these challenges by publishing international standards containing information on various aspects of generic and medical device software testing. However, a review of these standards has revealed a lack of a consolidated set of requirements in the form of a single standard which incorporates software testing best practice and related development and risk management activities that are required for the development of medical device software. This thesis addresses this lack of consolidated information through the development and validation of the software testing MED-V-STEP framework. This framework maps the activities of the most recent generic software testing standard to the relevant activities of the medical device software development and risk management standards and defines the relationships between them. The framework enables the implementation of software test processes to maintain these relationships between test, development and risk management activities.

The development of the MED-V-STEP framework generates a contribution by providing organisations with the knowledge of how to implement generic non-domain-specific test processes for testing medical device software to address identified software testing challenges and improve testing efficiency. The MED-V-STEP framework has been validated by focus groups and questionnaires carried out with medical device software development organisation and software testing organisation. The validation findings indicate the benefits of testing using the MED-V-STEP framework rather than multiple standards. The framework was validated as improving the software testing efficiency and enabling implementation of software testing best practice for medical device software with low implementation overhead.

Map of Thesis – Part 1



Chapter 1 Introduction

The potential of software to provide increased functionality has resulted in the development of increasingly complex and extensive software in various industries over the past few decades. Using such software offers many benefits but can also bring issues related to quality and safety. Ensuring software quality poses a significant challenge due to the intangible nature of software. Software development involves software testing and other verification & validation activities to prevent or detect defects that may reduce its quality or introduce a safety risk to users. This is evident in the healthcare field, in which the software functionality improves the services provided by medical devices, but at the same time may introduce safety problems such as injury to or death of patients. Due to the significant impact of software on the functionality of medical devices and the services they provide, and the existing software quality and safety issues, this thesis focuses on the role of software testing in terms of software quality and safety, and the challenges related to software testing and improving medical device software testing.

Section 1.1 in this chapter outlines the role of testing generic software and medical device software as well as problems that exist in terms of software quality and safety. Software testing challenges faced by software development organisations in addressing the quality and safety issues are presented in Section 1.2. Section 1.3 provides an overview of the role of standardisation organisations and international standards in software process improvement and how they contribute to the improvement of generic software testing and medical device software testing. This is followed by Section 1.4 outlining the design and development of an innovative software testing framework to address the challenges associated with medical device software testing. Section 1.5 specifies the research questions to be answered in this thesis, while Section 1.6 specifies the research objectives to address these research questions. Section 1.7 outlines the contributions generated by this thesis. The document structure with an overview of the following chapters is presented in Section 1.8.

1.1 Software Testing

Testing is indispensable in software development to determine software quality (Planning 2002, Bertolino 2007). The process of running the software against a number of tests and evaluating the test results is the most frequently used means for assessing software quality (Tian 2005 p.5). Human error in the development process poses a risk of defective and therefore poor-quality software, as some defects can lead to software malfunction (Graham et al. 2008 p.6). Therefore, one of the main intentions of testing is to detect defects so that they can be removed. The removal of defects raises the quality of software and prevents customer dissatisfaction, loss of revenue and reputation (Mayers 2004 p. 8).

Software development in the medical device domain has grown considerably within the last 20 years, with an increasing number of medical devices relying on software for their principal functionality (Lee and Sokolsky 2010, Rastgarpour and Shanbehzadeh 2011, Toussaint et al. 2007). The functionality of software embedded in medical devices is increasingly used for diagnostic or treatment purposes, as it improves the quality of the health services (National Cancer Institute 2015, Camara et al. 2015, Monti et al. 2010).

The growing use of software increases the requirements for software quality, and hence software testing (McHugh et al. 2013, Leveson 2000). Growing functionality increases the complexity of software and leads to the evolution of software development to deal with this complexity (ISO/IEC 2008, IEC 2015). Software testing becomes more complex as a result of an increasing occurrence of defects, the number of which increases due to increasing complexity of software systems and evolving models of software development process. In the medical device domain, software testing contributes to safety and reduces the risk of threat to humans due to defective software (Knight 2002).

A series of reports related to generic software quality issues demonstrate a noticeable trend of challenged and failed software development projects (The Standish Group 2014). There is also an increasing trend in medical device recalls due to software issues (Dix et al. 2016). A recent review reveals an increasing number of recalls of medical devices and adverse events to the patients due to software failures (SRCL 2018). The main causes of medical device software quality issues are inappropriate software development process and ineffective software testing (Vector Software 2016, Dix et al. 2016). Since there are quality issues in software development, including that of medical device software, the next section reviews the literature to identify the most prevalent software testing challenges in both generic and medical device domains.

1.2 Software Testing Challenges

In this thesis, a literature review identified challenges related to testing increasingly large and complex software systems. Growing software functionality increases software size and complexity, requiring more extensive software testing. (Lee et al. 2006). However, software testing must be conducted within a finite time frame, and for both generic and medical device software, available resources including time and money do not allow for complete testing (Vector Software 2016). Software testing represents 20% of the time and money spent on the entire development process for generic software, up to 50% for safety-critical software and still, the effective implementation of software testing is challenging (Hailpern and Santhanam 2002, King 2015). Due to these challenges, the emphasis is on improving the effectiveness of software testing in detecting as many defects as possible, especially those threatening software functionality (Reid 2013).

The need to address software quality has resulted in the requirement for standards to describe good development practices. Currently, there are a number of standards bodies that have published standards with the intent to improve software development (Clarke and O'Connor 2010). The software development life-cycle (SDLC) standards introduce various life-cycle stages from software development planning to software release and maintenance (ISO/IEC 2008 p.13). The risk of human error at the early life-cycle stages such as requirements specification or design specification brings another challenge to software testing. Incorrect or incomplete requirements specifications are expensive to correct when identified at the software testing phase. If, however, human error could be identified by the software testing activities at early life-cycle stages, it would make the correction 100 times less expensive (Langenfeld et al. 2016, Seth et al. 2014). Efficient implementation of software testing covering early life-cycle stages is a challenge for software development organisations and software testing related standards play a role in the improvement of its efficient implementation (Gelperin and Hetzel 1988).

Greater emphasis on safety in the medical device domain increases the significance of software testing with respect to the extent of testing (Lee et al. 2006). Providing high-quality and safe software is a high priority for medical device software development organisations and is subject to regulatory requirements (European Union 2017). The compliance of software development is best achieved with the use of the standard IEC 62304 which is harmonised with medical device regulations (European Commission). The importance of harmonised standards for the compliance of medical device software development is discussed in the following section.

1.3 Standardisation Organisations & International Standards

Standardisation organisations publish international standards that provide recommended practices for various professions in various domains (ISO, IEC, IEEE). Compliance with any standard requires the fulfillment of all processes and other requirements specified by it. If the international standard is harmonised with regulations, for example for the European Union or the United States, the fulfilment of standards requirements is considered as compliance with relevant regulations (European Commission). If an organisation using a harmonised standard has been audited by a notified body designated by a government authority, and was awarded a mark of compliance with regulations, its product can be placed on the market on which it complies with the harmonised standard.

This thesis reviewed international standards related to generic software testing and quality (ISO 2018a) and software development in the medical device domain (ISO 2018b), which have the potential to contribute to solving software testing challenges in the medical device domain. According to the review's findings, information on generic software testing and harmonised medical device software processes is included in and dispersed among multiple standards. The most recent generic software testing standard has not been tailored for use in the medical device domain to be based on the requirements of harmonised medical device software standards.

Therefore, organisations face challenges in the identification of relevant testing standards and how their requirements relate to the processes of harmonised standards for medical device software. Implementation and compliance with medical device software development standard is seen by software organisations as a challenge. The need to ensure compliance of software testing with requirements of multiple international standards poses more significant challenges. In addition, the need to maintain relationships between these standards brings a further challenge. For example, software development activities, such as requirements analysis or design, are related to the test planning process because they generate the data as input needed for test planning activities.

Since software testing plays a significant role in improving the quality and safety of medical device software, and the sets of requirements addressing software testing challenges are dispersed between multiple standards, the following hypothesis is going to be tested in this thesis: *"An approach to consolidate generic software testing best practice with related requirements of harmonised medical device software standards has an impact in addressing identified software testing challenges"*. Testing this hypothesis will answer the Research Question presented in Section 1.5.

1.4 Towards a Software Testing Best Practice Framework

This thesis aims to provide a consolidated resource on medical device software testing through the development a novel software testing framework based on the standards indicated in Section 1.3. The selection of relevant standards is addressed in the literature review section. The ISO/IEC/IEEE 29119-2 standard provides generic software testing best practice that addresses the challenge of detecting an increasing number of defects in software systems that are increasing in size. The IEC 62304 medical device software life-cycle processes (SLCPs) standard and the ISO 14971 risk management standard contain information, which addresses the compliance of software testing with medical device regulations. The relevant information from these standards is used in this thesis for the development of a software testing framework with the aim to address identified software testing challenges, and to assist organisations in the efficient implementing of ISO/IEC/IEEE 29119-2 software test processes for medical device software testing.

The challenge of implementing the ISO/IEC/IEEE 29119-2 generic software testing standard in conjunction with medical device SLCPs and risk management standards resulted in the decision to consolidate information by identifying their relationships to each other. According to ISO/IEC/IEEE 29119-1 in describing software testing concepts and definitions, the test processes consist of interrelated and interacting activities (2013a p.13). According to IEC 62304, SLCPs describe the sequence of activities and tasks which are related to each other based on a logical dependence, where the output of one activity generates the input of another activity (IEC 2015 p.43).

Relationships with defined logical dependencies imply the activities are performed in sequence where the generation or change of the initial activity affects subsequent activities. For example, the *Identify & Analyse Risk* activity from ISO/IEC/IEEE 29119-2 must be conducted prior to *Identify Risk Mitigation Approach* activity, because the approach to mitigating risk can be identified based on risk analysis. Therefore, implementing activities in accordance with these relationships improves their efficiency. Developing a software testing framework that defines the relationships between activities that come from various standards eliminates the need for organisations to define how these activities are related and facilitates the implementation of software testing in accordance with these relationships. The research questions set out in Section 1.5 and research objectives in Section 1.6 are formulated to guide this study and to develop and validate an innovative software testing framework to answer these research questions.

1.5 Research Questions

The formulation of the Research Question (RQ) arises from the observation of an increasing number of recalls of medical devices and adverse events to the patients due to software failures, and how to prevent this through efficient application of software testing.

How can generic software testing best practice be implemented to take into account the requirements of medical device software life-cycle processes related to software testing in order to address the software testing challenges in the medical device domain?

The RQ addresses the fact that the defects causing software failure in the medical device domain were not detected by software testing and hence not removed. The primary focus of this thesis is the development and evaluation of a software testing best practice framework for medical device software to answer the RQ. The RQ was divided in four Research Sub-Questions (RSQ) as follows:

RSQ.1 What are the challenges associated with the implementation of generic and medical device software testing?

As software failures indicate undetected defects, RSQ.1 is related to challenges in the implementation of software testing. A literature review was conducted to identify challenges associated with generic and medical device software testing.

RSQ.2 How can the challenges associated with the implementation of software testing in the medical device domain be addressed?

RSQ.2 was formulated to investigate the techniques in improving software testing and whether international standards can contribute to addressing medical device software testing challenges.

RSQ.3 Can a framework be developed to implement software testing best practice while addressing the requirements of medical device software life-cycle processes related to software testing?

RSQ.3 is related to the development of a software testing best practice framework with the aim to contribute to the implementation of generic best practice software testing for the medical device software.

RSQ.4 Can the use of the framework contribute to the implementation of software testing best practice for the medical device software?

RSQ.4 concerns the validation of the impact of the framework on addressing software testing challenges faced by organisations developing medical device software.

1.6 Research Objectives

Since the research focus is to assist manufacturers in the implementation of software testing best practice in medical device software development, the following Research Objectives (RO) have been specified with the aim to design and develop a framework to assist in this focus and answer the RSQs and overall RQ:

RO.1 Investigate the challenges associated with the implementation of generic and medical device software testing.

RO.1 aims to gain an understanding of the importance of testing software quality, the differences of medical device software testing compared to generic software testing, and the generic and medical device domain-specific software testing challenges. These challenges are investigated through a literature review in Chapter 2 of this thesis. An examination of the challenges in generic software testing is reported in Section 2.2, while the challenges in medical device software testing are provided in Section 2.3.

RO.2 Determine techniques that can be used to address the challenges associated with the implementation of medical device software testing.

RO.2 aims to determine what techniques are relevant for the improvements in medical device software testing and software quality assurance areas as well as addressing the identified challenges in RO.1. A review of international standards was conducted to examine the utility and limitations of standards in addressing software testing challenges. An overview of the international standards for software testing and quality assurance, as well as the standards for medical device software, and the results of this review are given in Section 2.4.

RO.3 Develop the framework which incorporates software testing best practice while addressing the requirements of medical device software life-cycle processes related to software testing.

The aim of RO.3 is to design and develop the software testing framework based on conclusions from the literature and standards review carried in RO.1 and RO.2. The framework should consist of standards' processes and requirements related to software testing of medical device software. Framework design and development by mapping to identify relationships between selected standards is described in Chapter 4. This chapter also introduces the developed MED-V-STEP framework and gives examples of related requirements from different standards, as well as their logical dependence.

RO.4 Validate the framework in terms of providing information on the implementation of software testing best practice for the medical device software.

RO.4 aims to gain an evaluation of the developed MED-V-STEP framework by professionals from medical device software development and software testing industry on the overall structure and the usefulness of the framework for the implementation of software testing best practice in medical device software development. Carrying out the focus groups and questionnaires, as well as the evaluations and views obtained on the usefulness of the MED-V-STEP framework, are described in Chapter 5.

The four ROs address the corresponding four RSQs and contribute to answering the overall RQ regarding the software testing challenges, the relevant techniques to address these challenges, the development of a software testing best practice framework, and the validation of the usefulness of the MED-V-STEP framework for implementing this best practice.

1.7 Research Contributions

In response to the research objectives identified above, this thesis delivers the following key contributions:

- Software testing challenges faced by medical device software organisations in ensuring software safety and quality have been identified.
- Relevant international standards and their suitability to address these challenges have been determined. At the same time, a limitation of their use was identified in that there is no consolidated resource in the form of a single standard that addresses all challenges.
- An innovative software testing framework named MED-V-STEP has been developed to help organisations implement software testing best practice for medical device software.
- The impact of the framework on the implementation of software testing best practice for medical device software has been validated through focus groups and questionnaires.
- This thesis provides basis for future extension of the software testing framework and development of a medical device software quality assurance best practice framework to assist organisations developing medical device software in the implementation of software testing and other verification & validation best practices.

1.8 Document Structure

The thesis consists of six Chapters. Chapter 2 presents the literature review related to software testing and quality assurance challenges. This chapter also includes the standards review to examine how they can contribute to addressing these challenges.

The overview of research methodologies is provided in Chapter 3. This chapter describes the selection of a research design for this study and the justification for its suitability for developing and validating an innovative software testing framework.

Chapter 4 presents the approach to the design and development of the software testing best practice MED-V-STEP framework for medical device software. This chapter describes the mapping performed in this study, which identified software testing related requirements dispersed between various standards, and how the mapped requirements were defined in terms of their relationships to each other. The developed framework based on the performed mapping and defined relationships is presented in this chapter.

The validation of the framework through the focus groups supported with questionnaires is discussed in Chapter 5. This chapter presents the focus groups that were performed with the Medical Device Software Development Organisation and the Software Testing Organisation and provides the findings from these focus groups.

Chapter 6 provides a final summary and conclusions drawn from the research. This chapter also presents research contributions, limitations and outlines future work.

Chapter 2 Literature Review

2.1 Introduction

This chapter reviews the literature in the field of software testing and verification & validation. The aim of this chapter is to provide an understanding of the role of these activities and related challenges in the development of generic and medical device software. Section 2.2 describes how the changes in software development triggered the evolution of software testing into software quality assurance consisting of verification & validation. This section also identifies the challenges of generic software testing. The review of the literature of software testing in the medical device domain is presented in Section 2.3. This review was conducted to determine differences between this safety-critical domain compared to generic software testing and identify software testing challenges specific to the medical device domain. Section 2.4 focuses on exploring what techniques can be used in software process improvement and examining international standards for their relevance in improving software testing. This section reviews international standards to identify those related to software testing, verification & validation and medical device software development and assesses to what extent they can address the identified generic and medical device software testing challenges. This section also presents existing approaches to the joint use of multiple standards through standards integration and consolidation and provides conclusions. Section 2.5 revisits research questions and objectives and surveys and critiques the presented literature review. The final Section 2.6 outlines an approach for the development of a novel software testing best practice framework for the medical device software.

2.2 Generic Software Testing

2.2.1 The Purpose of Software Testing

In the software development process, software testing is a critical phase for assuring software quality and is inherent to both software engineering and quality assurance (QA) (Bertolino 2007). One of the primary goals of software testing is to address errors made by software engineers (Tian 2005 p. 5). Error is a human action that produces an incorrect result such as software containing defects (Tian 2005 p. 20). A defect is a flaw in a software component or system that can cause the component or system to fail to perform its required function (ISTQB 2016).

The process of running software in a development environment based on sets of tests and evaluating test results is the most frequently used way to assess software quality (Tian 2005 p. 5). Software testing achieves software quality objectives by identifying and removing defects. Software testing can provide confidence in software quality when the results of test execution are consistent with the specification of a tested software system (Whittaker 2009, Chillarege 1999). When the test results demonstrate software inconsistency with the specification, then the quality of software needs to be enhanced. This is achieved by detecting and eliminating the defects that are the cause of the functional inconsistencies demonstrated by test results (Myers 2004 p. 10). The minimisation of the severity of defects raises the software quality and decreases software development costs (Meenakshi et al. 2014). Software testing addresses engineers' errors in software development and defects in software functionality. Section 2.2.2 describes how the increasing functionality contained in software and the evolving development processes have influenced the evolution of software testing into its current state.

2.2.2 The Evolution of Software Testing

Software testing has existed as a separate testing activity since programming began and has been a fundamental part of software development before software development life-cycle (SDLC) models were defined (Reid 2013). In 1950, the article *Computing Machinery and Intelligence* was published that could be considered as the first on software testing (Turing 1950). Over the decades, the growing functionality provided by software has made many software systems highly complex containing millions of lines of source code (Charette 2009, Tian 2005 p.4). SDLC models have evolved to meet the development needs of such software systems. Software testing has been changing along with increasing complexity and magnitude of software systems and evolving SDLC as presented in the following subsections.

2.2.2.1 Software Testing in the Software Development Life-Cycle

The demand for increasingly sophisticated software with a growing number of lines of code resulted in changes to software development life-cycle (SDLC) models. There are a number of existing SDLC models, such as waterfall (ISO/IEC 2008 p. 13), incremental (Mujumdar et al. 2012), evolutionary (ISO/IEC 2008 p. 13), as well as others, which are available for the development of various types of software. The common characteristics of these models are identifiable stages such as planning, requirements specification, design, implementation, testing and maintenance, inputs to each stage, transformation of input into output within the stage and outputs of each stage (ISO/IEC 2008 p. 10).

Growing functionality and increasing complexity and magnitude of software systems along with evolving SDLC models resulted also in changes to software testing. Well-focused and well-designed rigorous testing is necessary to identify defects and provide confidence in the quality of these complex software systems. The need for more rigorous testing resulted in an evolution of software testing from a single sequential activity affecting just software implementation into a process consisting of a set of test activities (Graham et al. 2008 p. 13). The term *software testing* or *testing* used previously in relation to the software implementation now takes on a broader meaning which includes activities such as test planning, test design, test implementation and test execution (ISO/IEC/IEEE 2013a p.19). Some test activities have shifted from the end to the beginning of the SDLC and are on a parallel track affecting requirements and design specification as well. These changes to software testing aim to provide well-designed and focused testing detecting defects as early as possible (Seth et al. 2014).

2.2.2.2 Evolution of Software Testing into Software Quality Assurance

In addition to software testing, there are other sets of activities to detect defects, such as reviewing the requirements specification, design review, or software analysis. By including these activities, testing has evolved into software quality assurance (SQA) often referred to as verification & validation (V&V) (Note 2014, Bertolino 2007). Verification refers to the set of activities that ensure that the software correctly implements a specific function, while validation refers to another set of activities that ensure that the software is traceable to customer requirements and intended use (ISTQB 2016, ISO/IEC 2008). In other words, SQA consisting of V&V provides confidence that correctly specified requirements have been implemented correctly and meet requirements specifications and intended use of software. In this way, SQA builds confidence in the reliability and effectiveness of the software to complete its specified tasks. Figure 1 which is an

adaptation of Vogel's diagram, presents SQA consisting of V&V and software testing and outlines the existing relationships between these activities.

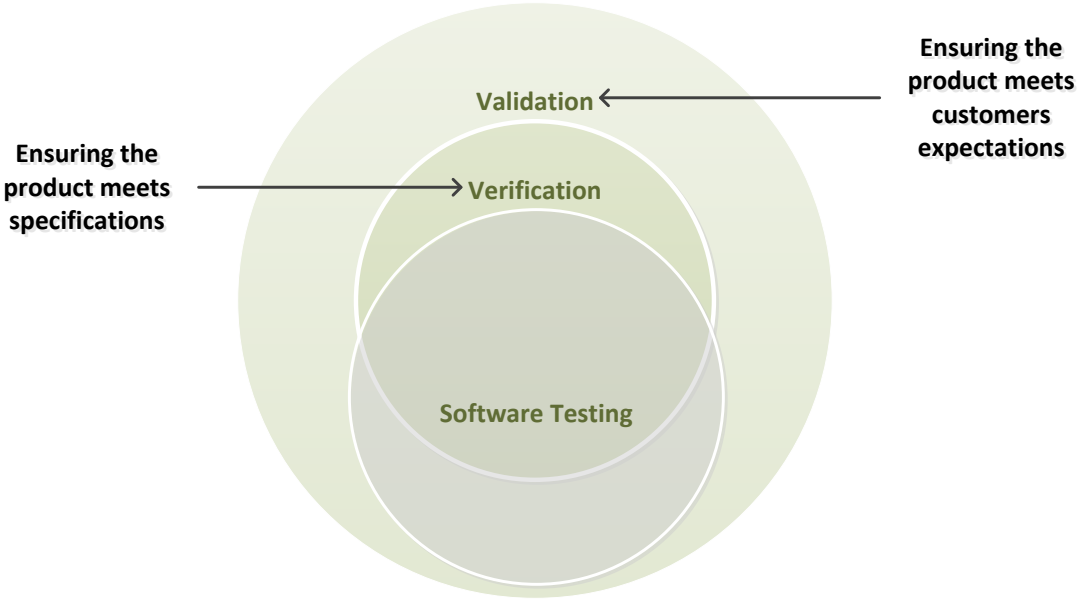


Figure 1 SQA Encompassing Verification & Validation - adapted from Vogel (2011 p. 77)

As illustrated in Figure 1, software testing is related to both, V&V. According to Vogel, verification covers most of software testing activities. Non-verification software testing such as user preference testing or ad hoc testing is covered by validation. Although this diagram is related to the medical device software domain, it is consistent with how ISO/IEC/IEEE 29119-1 describes software testing as a part of V&V (2013a p. 35). The adaptation made in this diagram applies to different views on the scope of software testing within V&V. While Vogel believes that most V&V activities are non-testing activities that overlap with software engineering good practices (Vogel 2011 p. 77) and software testing is represented in his diagram by a small ring, this diagram takes into account the view presented in ISO/IEC/IEEE 29119-1 and other literature that software testing represents the majority of V&V (2013a, Bertolino 2007).

The importance of software testing for software quality and the requirement for verification software testing to achieve compliance with medical device regulations direct the focus of this study on medical device software testing within verification activities which is presented in Section 2.6. Section 2.2.2 described the evolution and current state of software testing, while the following section presents the challenges related to software testing of generic software systems.

2.2.3 Challenges of Generic Software Testing

Software testing challenges related to generic software systems are not related to one specific domain but may appear in various domains regardless if they are safety-critical or not. In this study, the term *software system* is defined as an integrated set of software components organised to accomplish a specific function or set of functions (IEC 2015 p. 17).

2.2.3.1 Defects due to Increasing Magnitude and Complexity of Software Systems

The increasing magnitude and complexity of software systems pose challenges and cause the evolution of software testing to cope with these challenges (Seth et al. 2014, Hastie and Wojewoda 2015).

Software quality issues due to software magnitude have been highlighted by a survey conducted on the rate of successful and failed software projects in all domains in 2013 to 2017 (Mersino 2019). The results of the survey reveal that smaller software projects, compared to the larger ones, tend to achieve lower levels of failure and provide better software quality. For many large software projects, there is a significant problem with providing high-quality software development in a repeatable and sustainable manner (Avgeriou et al. 2015). A software system with millions of lines of code could contain thousands of program-related defects, which poses a challenge for comprehensive testing of these software systems.

The Standish Group study on software project failures and challenges for the years 2011 and 2015 revealed that for the software system complexity, the same dependency applies as for software magnitude: with an increase in software complexity the risk of project's failure increases (Hastie and Wojewoda 2015). Other studies highlight the need to improve software testing, stating, that the pace of progression of software testing due to increasing software complexity is insufficient (Bertolino 2007, Myers 2004 p. 4).

One of the significant reasons for the ineffective software testing practice is that with increasingly extensive and complex software systems, due to constraints in terms of time and human resources, it is not feasible in practice to cover all quality aspects and to test software system exhaustively (Kaner 2003, Myers 2004 p.10, ISO/IEC/IEEE 2013a p.13). The scope for software testing of a software system is usually so huge that only a very small proportion can be tested and even achieving full-code coverage is challenging even for the best testing techniques (Jackson 2009). Software testing must therefore focus on the most important areas of software functionality to be covered (Reid 2013).

Some defects are a consequence of human errors committed during the software implementation, while others are introduced at other life-cycle stages, such as requirements specification or design. If they are not detected, they affect the subsequent stages and, consequently, reduce the quality of the software. The following subsection examines the software testing challenges related to the various life-cycle stages.

2.2.3.2 Defects Occurring at Various Stages of Software Development Life-Cycle

At the international level, standardisation organisations are involved in the evolution of SDLC processes with the aim to improve these processes, and thus the quality of the software under the development. As a result of this effort, international standards related to software engineering have been published providing requirements which need to be fulfilled and processes which need to be implemented to improve the software development practice. As an example, one of the widely known and used standards for generic software development is *ISO/IEC 12207 Software life-cycle processes* published in 1995 and amended in 2008, providing requirements for the implementation of software development best practice (1995, 2008).

The evolving SDLC models introduce various life-cycle stages aiming for software process improvement which, however, become a potential new source of human error. Employees involved at any stage of the SDLC introduce human fallibility and the risk of committing errors resulting in defects. Undetected defects that appeared in the early life-cycle stages affect subsequent stages. The later they are detected, the more expensive it is to remove them. An example of this is an analysis of a five-year observation of requirements defects in the automotive domain which identified human errors committed at the requirements specification stage resulting in incorrect, inconsistent or incomplete requirements (Langenfeld et al. 2016). A lesson learned from this observation was to include test engineers in the project from the beginning in all requirements reviews to improve the efficiency in the identification of requirements defects.

Critical review of requirements and designs can find defects before implementing defective requirements or designs (Chillarege 1999). The cost of correcting such defects early in the SDLC is from 10 up to 100 times less expensive than it is late in the SDLC (Chillarege 1999, Vogel 2011 p. 61). Software test activities such as test planning, analysis or design, use the output of development activities, such as requirements specification or software design. These testing activities enable testers to detect incompleteness, ambiguity, inconsistency, and incorrectness of requirements and design specification (Gelperin and Hetzel 1988 p. 687). The shift of test activities to the

beginning of the SDLC is beneficial for rigorous planning and design of software testing but also in detecting and removing defective requirements and design specifications at an early life-cycle stage (Planning 2002). A research on challenges related to software testing affecting the quality of software was carried out in software development companies from various domains (Seth et al. 2014). According to this research, the implementation of test activities at the early life-cycle stages is not commonly practised and software testers were insufficiently involved in planning stage and some other life-cycle stages.

2.2.4 Summary & Conclusion

There are software testing challenges in detection of an increasing volume of defects and in the coverage of increasingly extensive and complex software systems. Industry software testing practices are still very often ineffective in ensuring high quality of extensive and complex software systems and to deal with related challenges. Smaller software projects, compared to the larger ones, achieve better software quality. The ineffective industry software testing demonstrates the need for the more efficient implementation of software testing. This can be achieved by implementing rigorous, well-focused and designed software testing.

Another generic software testing challenge presented in Section 2.2.3 relates to evolving SDLC models. Various life-cycle stages introduce risk of human errors committed at these stages, resulting in software containing defects. In response to this, software testing activity has been changing along with evolving SDLC models and evolved into a test process consisting of a set of test activities. Some of these test activities have moved to the early life-cycle stages such as planning, requirements specification and design.

Software testing has further evolved into SQA, which is often referred to as V&V. In addition to software testing, V&V includes other non-testing activities such as reviews and analysis. These activities also moved the testing focus to the beginning of the SDLC. The changes regarding the position that software testing takes in the SDLC, and the broadening of the software testing perspective, enable the identification of errors and defects at the early life-cycle stage reducing development costs. The next section considers whether the challenges identified in this section exist in the medical device domain and whether there are any additional software testing challenges specific to the medical device domain.

2.3 Medical Device Software Testing

As the focus of this study is on the medical device software testing, the following sections discuss medical device software testing challenges and outline differences in software testing of medical device software compared to generic software testing.

2.3.1 Software Testing Challenges in Medical Device Domain

A medical device software system is:

"a software system that has been developed for the purpose of being incorporated into the medical device being developed" (IEC 2015 p. 16).

According to the *2016 Software Testing Technology Report*, development of safety-critical software embedded in devices or machines such as the pacemakers or electronics of cars, also faces an increase in the size and complexity, making traditional development and software testing insufficient to deal with these growing challenges (Vector Software 2016). The growing functionality of medical devices increases the complexity and magnitude of software, and therefore the risk (McHugh et al. 2013). Consequently, software testing of medical device software becomes more complex and challenging for manufacturers (Lee et al. 2006). It is practically impossible to cover all quality aspects of increasingly complex medical device software systems, and complete software testing is not feasible in terms of time and human resources (King 2015, Vogel 2011 p. 27). These findings from the literature review revealed that the generic software testing challenges presented in Section 2.2.3 exist also in medical device software testing.

However, in safety-critical domains such as medical device, automotive or aerospace domains, software defect resulting in a malfunctioning device can lead to catastrophic consequences and pose a threat to human life or the environment (Steinbrook 2005). In the medical device domain, the increased occurrence of defects increases not only the risk of software failure but also the risk of harm to the patient, clinician or third party (McHugh et al. 2013). Due to the substantial impact of a software defect, which can cause loss of life or significant injury, there are safety requirements aiming for the development of reliable and safe medical device software (Lee et al. 2006, Knight 2002). Defective software, when embedded in medical devices, could be the subject to withdrawal, reimbursement and consequently loss of profit (Dix et al. 2016, ISO/IEC/IEEE 2013a p. 13, Wallace and Kuhn 1999). The purpose of software testing is to minimise or eliminate such adverse consequences of errors caused by employees involved in medical device SDLC (Tian 2005 p. 5). For safety-critical systems, the literature confirms the need for

prioritising of test activities to address the safety areas with the highest criticality and covering all life-cycle stages with rigorous software testing (Heimdahl 2007). The following Section 2.3.2 describes the regulatory situation applicable to medical device software development and software testing.

2.3.2 Medical Device Regulatory Requirements

Tragic events related to the use of malfunctioning medical devices led to attempts to address the safety issues through regulatory oversight. The safety issues related to medical device development became a matter of legal concern for national and international authorities. United States (U.S.) Health authorities introduced into the legislation the Safe Medical Device Act of 1990 which mandates reporting requirements by medical device manufacturers regarding adverse safety events (Food and Drug Administration 1990). They also published the 1996 Quality System Requirements which specified the quality system requirements for the design and development of medical devices (Food and Drug Administration 1997). Next, between the years 2010 and 2015 almost two hundred draft and final Food and Drug Administration (FDA) guidance documents related to the medical device SDLC management were published (Dix et al. 2016).

As software increasingly was used in medical devices, medical device software development as a safety-critical domain became subject to regulatory requirements which apply to both, embedded or standalone software (European Union 2017). The regulatory amendment issued by European Commission in 2010 changed the classification of software meaning that software used for treatment and diagnosis as per the established definition of the medical device now could be classified as a medical device in its own right (2016). Recently, the European Commission issued guidance on qualification and classification of medical device standalone software (2016).

National government bodies took regulatory oversight of medical device development and issued regulatory requirements aiming to raise the safety of medical devices. The national regulations of European countries are based on the regulatory framework provided by the European Union (EU) Council, and in the U.S. by the Federal Government (McHugh et al. 2011). By these authorities, the European Union Regulations on Medical Devices (European Union 2017), and the U.S. Code of Federal Regulations were issued (U.S. Government Publishing Office 2016). If a product or service complies with regulatory requirements, a certificate is issued, which entitles the organisation to sell products on the market (Food and Drug Administration 2018, European Union 2017).

2.3.3 Medical Device Software Safety

Despite the regulatory effort on the international and national level, there is still evidence of adverse events and recalls of medical devices related to software issues. The continuous upward trend of adverse events, especially over the last fifteen years, demonstrates that safety and quality problems continue to grow and need an urgent solution (Simone 2013).

An analysis on medical device recalls due to software issues based on the U.S. FDA data of medical device malfunctions and recalls, reveal that for the year 2006, there was evidence of 4,500 deaths and 116,000 injuries due to said malfunctions (Dix et al. 2016). The analysis also demonstrates that from 2010 to 2015 there has been an upward swing in the following root-causes: software design – 96%, software change control – 214%, and software manufacturing process design – 460% increase. The last root-cause with the highest increase is related to the SDLC including software testing and demonstrates a significant need for improvement in software development and testing.

In 2011, further research on U.S. FDA data stated that malfunctioning medical devices are one of the leading causes of severe injury and death in the U.S. According to this research, in 2011, 92,600 patient injuries and 4,590 deaths were reported to the U.S. FDA, and over 20% of these numbers are associated with computer-related adverse events (Alemzadeh et al. 2013). Also, according to this report, between 2006 and 2011, there was an upward trend both in the case of recalls and adverse events; the number of computer-related recalls of medical devices increased by 70%, and the number of computer-related adverse events to the patients increased by over 100%. During this period, the total number of computer-related recalls reached 1,210 which affected 12,024,836 devices, and 64% of them were due to software failures. This data demonstrates high instances of malfunctioning software in reported recalls and adverse events and indicate the urgent need for addressing safety and quality issues in medical device software development (Selwood 2012).

An example from 2016 is a demonstration of the continuing safety issues. A research team hacked into ten different types of implantable medical devices and pacemakers so that they could affect their functioning remotely without physical access to the device (Pauli 2016). Such intentional action could endanger the safety and lives of patients. The vulnerability of the medical device was proved by a black-box approach to listening to the wireless communication channel and by reverse-engineering the proprietary communication protocol. This experiment indicates ineffective recognition of a

predictable risk such as the vulnerability of a device for hacking. Literature review findings suggest that if software testers used the requirements specification for test planning and design, they could detect missing safety requirements and prevented the vulnerability of medical device software (Gelperin and Hetzel 1988).

The *U.S. Recall Index* revealed that medical device recalls have increased by 126% to 343% in the first quarter of 2018, more than double the last quarter. Malfunctioning software was the reason for 23% of medical device recalls, which is the highest percentage of all causes of the recalls (SRCL 2018). This figure demonstrates the continuous upward trend and the need for a solution addressing software quality issues and related software testing challenges.

2.3.4 Summary & Conclusion

Section 2.3.1 revealed that generic software testing challenges due to the increasing magnitude and complexity of software systems, in detecting the increasing volume of defects and due to human errors committed at various life-cycle stages, also exist in medical device software development. Therefore, the same approach of well-focused and well-designed rigorous testing can be used to address these challenges. Government bodies attempted to address safety issues related to the use of medical devices by issuing relevant regulatory requirements. Section 2.3.2 identified that the software testing as a part of medical device SDLC has to comply with medical device regulatory requirements.

The findings of the literature review revealed that both software testing activities and medical device regulations are designed to provide high quality and safe software. Software testing aims to improve the quality and safety of software by detecting defects and regulations aim to ensure safety of software by regulatory oversight of the software development process. Despite this effort, the most recent data presented in Section 2.3.3 reveal the upward trend of adverse safety events and recalls of medical devices. The identified root-cause with the highest increase is related to the design and implementation of the SDLC which includes software testing. Therefore, the following Section 2.4 focuses on techniques for software process improvement and examines international standards for their relevance in improving medical device software testing.

2.4 International Standards Related to Software Testing & Quality and Medical Device Software Development

2.4.1 Introduction

For many years, international standards and publications related to software engineering played a significant role in the progression of software testing (Gelperin and Hetzel 1988). Knowledge of these generally accepted quality practices was recognised as necessary for effective use of software testing in medical device software development and addressing software testing challenges (Wallace and Kuhn 2001). Over the past few decades, standardisation organisations published standards with process and document specifications to improve software testing practice in the industry. Examples of the worldwide organisations are the International Organisation for Standardization (ISO), International Electrotechnical Commission (IEC) and Institute of Electrical and Electronics Engineers (IEEE) (ISO, IEC, IEEE). Although these organisations differ in the fields of their focus, there are some common areas related to software engineering and some of the standards are unified and published jointly by these organisations. Since international standards play a significant role in software process improvement, a review of international standards was conducted for their suitability to contribute to a solution for medical device software testing challenges identified in Section 2.3. The following sections provide an overview of international standards related to generic software testing, software quality and medical device software development.

2.4.2 Overview of International Standards

This study focuses on software testing in the medical device industry. Therefore, the standards have been reviewed and assessed for their suitability for use in medical device software testing. ISO standards are divided into groups based on a specific area that is dealt with by a specific technical committee. The following sets of standards from different technical committees were selected as related to the identified software testing and SQA challenges (ISO). The initial review examined the extent of utility of these standards in addressing the challenges associated with medical device software testing.

The first set is from technical committee *ISO/IEC JTC 1/SC 7 - Software and systems engineering* (ISO 2020a) presented by blue fields in Figure 2. This joint technical committee provides standards developed and published jointly by ISO and IEC for the software and systems engineering that is relevant to this research. Specific standards coming from this committee, which are the focus of this study are those related to software testing and software quality.

Another set is from technical committees *ISO/TC 210 - Quality management and corresponding general aspects for medical devices* (ISO 2020b) and *IEC/SC 62A – Common aspects of electrical equipment used in medical practice* (IEC 2020). For medical device software, regulatory oversight refers to the software development process (European Union 2017 p .5, IEC 2015 p. 10). From the standards developed by the *ISO/TC 210* committee, for medical device software development, there are standards harmonised with medical device regulations, which need to be implemented by software organisations to achieve certification (European Commission). The phrase *standard harmonised with regulatory requirements* means that fulfilment of standards requirements is considered as compliance with relevant regulations. The *IEC/SC 62A* committee developed standards covering the safety of health software products and the basic safety and essential performance of medical electrical equipment. The standards from these two committees are related to the study because they deal with medical device safety and SDLC stages including software testing as an inherent part. These standards are presented by green fields in Figure 2.

The next set was published by IEEE organisation and the U.S. Food and Drug Administration (FDA) presented by grey fields in Figure 2. This set covers verification & validation activities in both, generic and medical device domains and has therefore been identified as contributing to the medical device SQA.

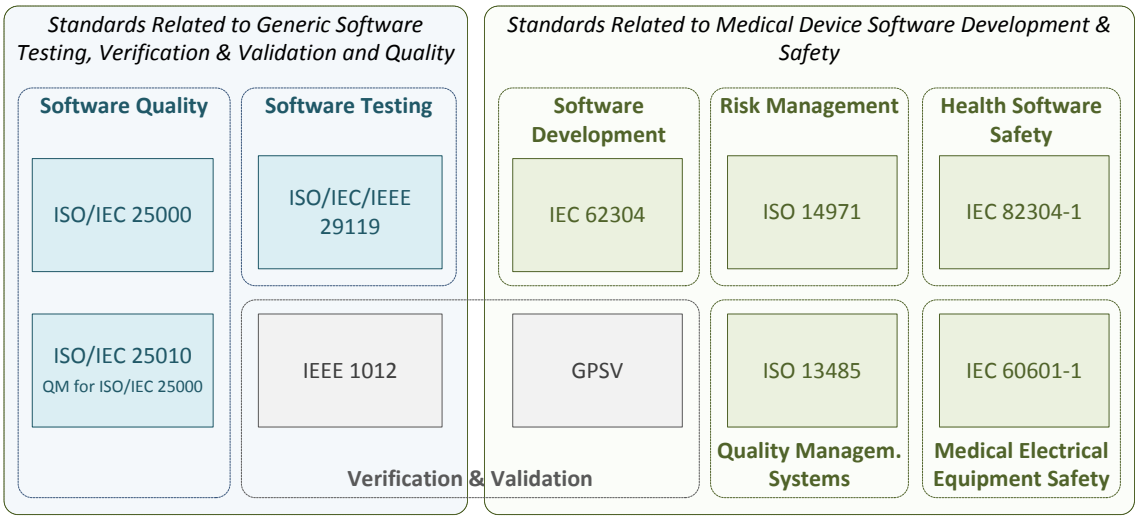


Figure 2 International Standards related to Medical Device Software Testing & SQA

All standards in Figure 2 have been identified as contributing to software testing and SQA best practice for medical device software, and they are therefore examined in the following sections, to which extent they contribute to addressing the medical device software testing challenges identified in Section 2.3.

2.4.3 Standards from Technical Committee ISO/IEC JTC 1/SC 7

Recently, the standard community has noticed the issue that software testing was not evolving fast enough compared to evolving SDLC models. In response to this issue, the set of international standards related to software testing and quality assurance has been published. These standards provide the most current software testing and software quality process models addressing the insufficient evolution of industry software testing (Reid 2013). The description of the extent to which the requirements of these standards contribute to addressing the challenges associated with generic software testing and software quality is provided below.

ISO/IEC/IEEE 29119-2:2013 Software and systems engineering – Software testing defines a generic process model for software testing consisting of sets of interrelated clauses that transform inputs into outputs. For the purposes of this study, terms defined in ISO/IEC/IEEE 29119-2 are further used to describe software testing as consisting of test clauses at different levels, such as test processes, test activities and test tasks. This standard defines the logical dependencies between related clauses as basis for their efficient implementation, where the output of one activity generates the input of another related activity. According to this standard:

> "*Testing is the primary approach to risk treatment in software development. This standard defines a risk-based approach to testing, which is recommended to strategizing and managing testing that allows testing to be prioritized and focused.*" (2013a p. vi).

The prioritisation and focus of testing based on the criticality of the increasing software functionality ensures that risks with the highest priority are paid the highest attention during testing of increasingly complex software systems.

ISO/IEC/IEEE 29119-2 addresses challenges related to evolving SDLC models by providing requirements for test processes which are aligned with clauses of generic SLCPs of ISO/IEC 12207 (ISO/IEC 1995). This standard aligned test clauses with SLCPs clauses, starting from early life-cycle stages of SLCPs such as requirements analysis and design. ISO/IEC/IEEE 29119-2 can be used in conjunction with any development model, and therefore also with medical device SLCPs (ISO/IEC/IEEE 2013b p.vi). By providing current software testing best practice that is aligned with recommended development practice, this standard addresses the insufficient evolution of industry software testing.

ISO/IEC 25000:2014 Systems and software Quality Requirements and Evaluation (SQuaRE) covers two main processes: software quality requirements specification and systems and software quality evaluation. It includes a two-part quality model for aligning customer definitions of quality with attributes of the development process. This quality model determines SQA validation activities addressing challenges related to the complexity and extension of a software system (ISO/IEC 2014).

ISO/IEC 25010:2011 SQuaRE – System and software quality models standard can be used in conjunction with ISO/IEC 25000 and provides models that can be used to establish quality requirements, their criteria for satisfaction and the corresponding measures (ISO/IEC 2011).

2.4.4 Standards from Technical Committees ISO/TC 210 & IEC/SC 62A

The international standards related to medical device software development identified in Section 2.4.2 comprise the requirements for the SDLC model. The standards developed by ISO/TC 210 are required for compliance with medical device regulations and deal with development and safety issues. Since software testing and SQA are integral to SDLC, the identified standards include requirements and processes which are associated with software testing and SQA. These requirements need to be considered for the contribution to the improvement of medical device software testing and SQA and compliance with regulations. The standards developed by IEC/SC 62A provide requirements for safety of health software and medical electrical equipment. Below is a description of the extent to which requirements of these standards can address the medical device software testing and SQA challenges.

IEC 62304:2006+A1:2015 Medical device software – Software life-cycle processes (IEC 2015) as developed based on generic *Software life-cycle processes* of ISO/IEC 12207 (ISO/IEC 2008), and the majority clauses of IEC 62304 are mapped to the corresponding clauses of ISO/IEC 12207. IEC 62304 similarly to ISO/IEC/IEEE 29119-2 defines interrelated clauses at different levels, such as processes, activities and tasks, and this terminology is used for the purposes of this study. This standard is harmonised by the EU and the U.S. with medical device regulations and used as a benchmark to comply with these regulations. Organisations can demonstrate software development compliance with medical device regulations by fulfilling the requirements of this standard.

The IEC 62304 standard requires the application of related clauses in adherence with the logical dependencies between these clauses, which is considered valuable in providing high-quality software development. IEC 62304 does not require or prescribe any

particular SDLC model and leaves the choice to the manufacturer. However, the chosen SDLC model has to fulfil the requirements of IEC 62304.

Software testing, as an inherent part of the SDLC, has to fulfil relevant requirements of IEC 62304 to demonstrate compliance with medical device regulations. IEC 62304 expects that verification will be covered as part of each IEC 62304 activity but does not address validation. This standard provides requirements for each life-cycle stage and determines those stages which need to be covered by software testing and SQA. The IEC 62304 standard refers to the other standards described below and requires applying SDLC model that also meets the requirements of these standards.

ISO 14971:2012 Medical devices – Application of risk management to medical devices (ISO 2012) is required by IEC 62304 to also be implemented in order to comply with medical device regulations. This standard deals with processes for risk management (RM), primarily to the patient, but also to the operator, other persons, other equipment and the environment. These processes determine the set of risk-based activities addressing the challenges related to safety issues.

ISO 13485:2016 Medical devices – Quality management systems – Requirements for regulatory purposes (ISO 2016) specifies requirements for quality management systems including the design and development of medical devices. This standard generally describes verification & validation of medical devices.

IEC 82304-1:2016 Health software General requirements for product safety (IEC 2016) applies to the safety of health software-only product that is designed to operate on general computing platforms. This standard covers entire life-cycle of health software intended to be used for managing, maintaining or improving halth of individual persons, or the delivery of care.

IEC 60601-1:2005+AMD1:2012 Medical electrical equipment – General requirements for basic safety and essential performance (IEC 2012) includes safety requirements for programmable electrical medical systems in which software is a subsystem. These requirements apply to software which provides safety functionality.

2.4.5 Standards and Guidance Related to Verification & Validation

IEEE 1012 - Standard for System, Software, and Hardware Verification and Validation provides V&V processes not related to software testing and a set of tasks for each life-cycle stage. This set of tasks can be used in conjunction with software test processes of ISO/IEC/IEE 29119 standard (2016).

General Principles of Software Validation; Final Guidance for Industry and FDA Staff (GPSV) outlines general validation principles which are by the FDA applicable to the validation of medical device software (Food and Drug Administration 2002, Sivakumar et al. 2011). GPSV therefore addresses SQA challenges related to compliance with U.S. medical device regulations.

2.4.6 Discussion

Software testing, V&V and software quality standards provide best practice for generic software testing, V&V and SQA but do not define how to implement them based on the medical device regulatory situation. According to *ISO/IEC/IEEE 29119-1 Software testing: Concepts and Definitions* standard, testing is based on the laws, regulations and industry standards applicable to the industry in which the organisation operates (2013a pp. 15-16). In the medical device domain, software development has to satisfy the requirements of the medical device regulations which is best achieved using harmonised international standard IEC 62304. To be based on medical device regulations, software testing has to be adapted to the requirements of IEC 62304 regarding software testing (ISO/IEC/IEEE 2013a pp. 15-16). However, ISO/IEC/IEEE 29119-2 does not provide information and is therefore limited in defining how to base ISO/IEC/IEEE 29119-2 test processes on the IEC 62304 SLCPs. Similarly, the ISO/IEC 25000 and ISO/IEC 25010 standards do not address the relationships of their processes with IEC 62304 SLCPs.

The ISO/IEC/IEEE 29119-2 standard contains the annexed table mapping test clauses of this standard to ISO/IEC 12207 clauses. Since IEC 62304 was developed based on ISO/IEC 12207, and most of the IEC 62304 clauses are mapped to the relevant ISO/IEC 12207 clauses, available annexed mapping table can be used to derive relationships between ISO/IEC/IEEE 29119-2 and IEC 62304 clauses. The relationships of IEEE 1012 V&V processes with IEC 62304 processes are not provided in this standard, but the annexed table provides high-level mapping of ISO/IEC 12207 V&V clauses to IEEE 1012 V&V clauses. This available annexed mapping table can be used to derive relationships between IEEE 1012 and IEC 62304 clauses. The processes of these standards can be used to address the insufficient evolution of software testing and SQA and extend software testing and verification related requirements of IEC 62304. Therefore, in this study, they were considered relevant to contribute to dealing with software testing and SQA challenges.

On the other hand, standards related to medical device software development deal with challenges related to the compliance with medical device regulations. IEC 62304 requires

verification to be performed but does not provide detailed information on how it can be successfully achieved. IEC 62304 is limited in describing software test processes in detail, therefore, there is a lack of detailed information on the efficient implementation of medical device software testing. Since software testing of safety-critical systems represents up to 80% of the entire software development cost and has a significant impact on the quality and safety of the software, the lack of a detailed description of software testing creates a gap in the IEC 62304 SDLC model (Reid 2012). Dr Stuart Reid, convenor of ISO Software Testing Working Group: WG26, states that the standards for software development in safety-critical domains require software testing to be performed, but do not provide definitions of these processes (2012). Also, the processes of ISO 14971 do not provide guidance on software testing and SQA to be performed. Although there is some description of V&V in ISO 13485, but it is too general and does not provide a detailed description that allows efficient implementation of V&V and software testing to medical device software. Also GPSV lacks detailed description of validation activities and tasks (Sivakumar et al. 2011). Consequently, in standards related to medical device software development, there is no consolidated information on the efficient implementation of software testing and V&V best practice for medical device software. As a result, the approach to the testing medical device software remains a matter of the individual choice of software development organisation. This can have a potential negative impact on the effectiveness of software testing, resulting in poor software quality and safety issues.

Section 2.3 identified software testing challenges in the medical device domain. Relevant standards and their utility and limitations in addressing software testing challenges have been described in Section 2.4. However, implementing medical device software testing information from these standards is challenging. The issue is the distribution of the required information across multiple standards. The approach proposed in this study is to address this challenge through standards consolidation.

Standards consolidation identifies and integrates the required clauses from multiple standards, and additionally, identifies existing relationships - the logical dependencies between these clauses, as described in Section 4.4. To facilitate the implementation of software testing best practice for medical device software, the next subsection examines existing approaches to standards consolidation and whether these approaches can contribute to consolidating information distributed between multiple standards related to software testing of medical device software.

2.4.7 Existing Approaches to Standards Integration & Consolidation

ISO has seen that organisations experience challenges in implementing multiple standards and published guidance on the integrated use of management system standards (2008). This example applies to standards other than those presented in this study. However, it demonstrates that the integrated implementation of multiple standards is challenging, and the need to facilitate the use of multiple standards requires some solutions. This study also aims to address the challenge related to the integrated use of multiple standards.

Research on standards integration and consolidation for the medical device domain was conducted and the innovative framework MDevSpice was developed providing best practice for organisations developing medical device software (Keogh 2015). Based on standard mapping, another study has been conducted to integrate and consolidate multiple standards into a software development framework for safety critical domains (Bujok et al. 2017). The examples of existing approaches to standards mapping, integration and consolidation of required clauses and the development of a framework incorporating best practice can be seen as a model for the consolidation of information on the implementation of software testing and V&V best practice in medical device software development, which is the subject of this study.

2.4.8 Summary & Conclusion

Literature review findings indicate a significant contribution of international standards to software process improvement, but medical device software testing has not evolved quickly enough compared to other SDLC activities. The harmonised standards related to medical device software development have to be complied with. However, IEC 62304 is limited in its discussion on testing and therefore there is a lack of detailed information on the efficient implementation of software testing for medical device software. Therefore, this study tried to identify if there are standards available to enhance IEC 62304 and address the issues around the evolution of software testing in the medical device domain. Section 2.4 examined the role of international software engineering standards in the improvement of test processes and identified standards available to enhance IEC 62304 and address the issues regarding the evolution of software testing in the medical device domain.

Section 2.4 has demonstrated that the identified international standards have the potential to contribute to addressing the software testing challenges identified in Section 2.3 and improving the software testing and SQA practice in medical device software development. These international standards are related to generic software testing, V&V

and software quality best practice, along with standards related to medical device software development and compliance with medical device regulations.

This study used information from multiple standards to address the insufficient evolution of medical device software testing. The implementation of multiple standards in a consolidated manner is a challenge for organisations. The existing approaches of the standards community and researchers to the mapping, integration and consolidation of standards into the framework aim to address this challenge. In Section 2.5, the Research Questions and Objectives are revisited, and the literature and standards review summarised.

2.5 Research Questions and Objectives Revisited

RSQ.1 What are the challenges associated with the implementation of generic and medical device software testing?

RSQ.1 is addressed by *RO.1 Investigate the challenges associated with the implementation of generic and medical device software testing.*

A review of the literature dealing with software quality issues was performed to identify the generic software testing challenges. As presented in Section 2.2, industry software testing is often ineffective in detecting an increasing number of defects and covering all quality aspects of software due to increasing complexity and magnitude of software systems and evolving SDLC models. Literature related to software development in the medical device domain was reviewed to examine the software testing challenges in this domain, which are presented in Section 2.3. According to the literature review findings, the generic software testing challenges also exist in the medical device domain. However, undiagnosed defects in medical device software can have a much greater impact than in generic software and can pose a risk to patients' safety or life. The software processes required to comply with medical device regulations do not describe software testing in detail, and test activities are based on manufacturer selection. The current state of medical device software development demonstrates increasing safety issues and indicates the need for the more effective software testing in the medical device domain.

RSQ.2 How can the challenges associated with the implementation of software testing in the medical device domain be addressed?

RSQ.2 is addressed by *RO.2 Determine techniques that can be used to address the challenges associated with the implementation of medical device software testing.*

In Section 2.4, a review of software engineering standards was performed to identify standards that can contribute to addressing software testing and SQA challenges identified in Sections 2.2 and 2.3. At the same time, an issue with the implementation of information distributed across multiple standards was identified. Existing approaches to the integration and consolidation of standards, which can be used as an example for the development of the software testing best practice framework for medical device software, were also presented. Section 2.6 outlines an approach to the development of the framework based on the integration and consolidation of required information from identified standards.

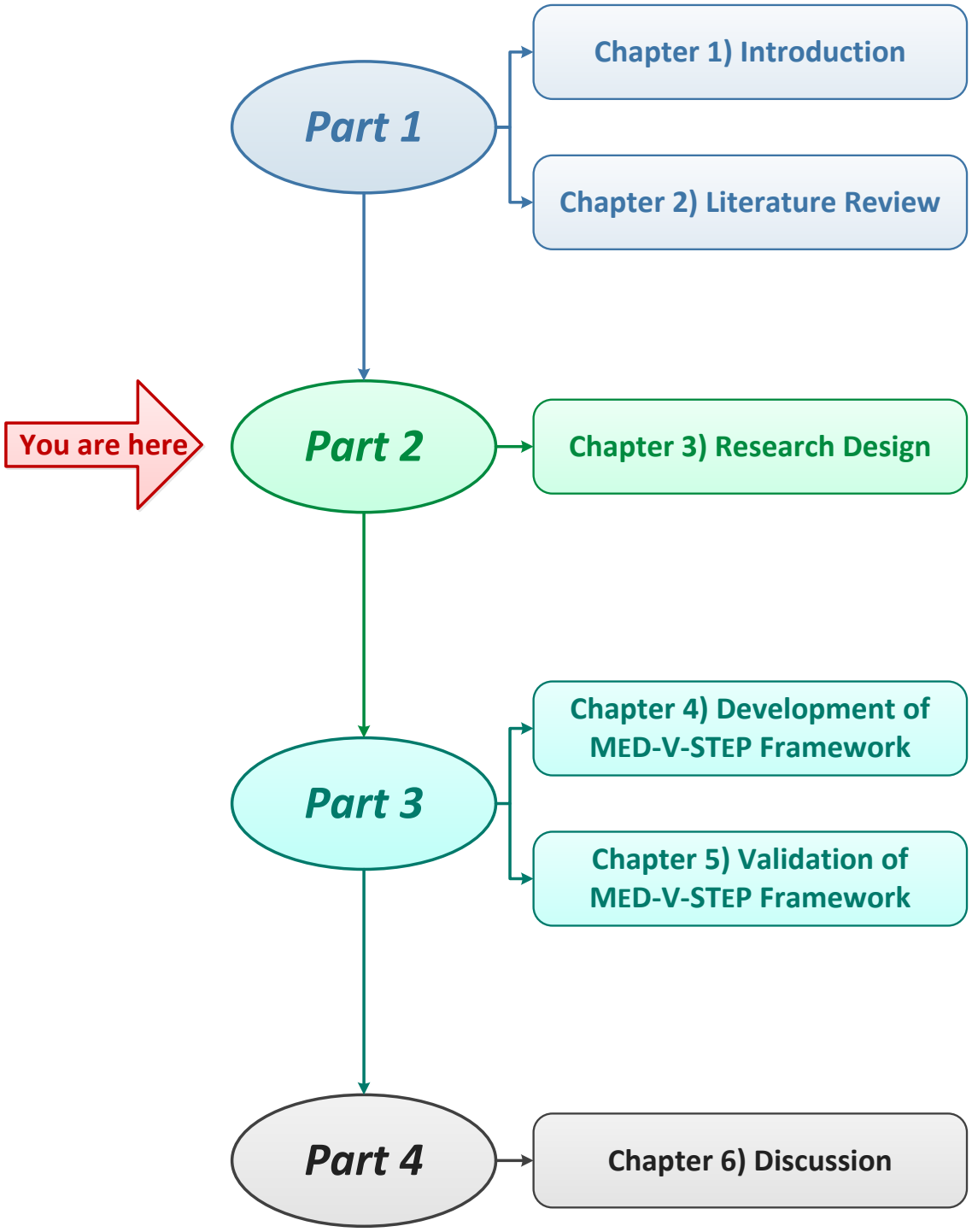
2.6 Approach to the Development of the Framework

The lack of guidance on software testing in medical device software standards has been identified as a research problem which needs to be addressed. The proposed solution is the development of an integrated and consolidated software testing best practice framework for medical device software. The framework was developed by identifying, collecting and consolidating information on testing medical device software from relevant international standards identified in Section 2.4. The scope of the framework undertaken in this study aimed first to address software testing challenges identified by the literature review in Sections 2.2 and 2.3. This framework aims to improve the practice of medical device software testing by assisting organisations in implementing software testing best practice based on requirements of medical device SLCPs and related to them.

A strategic decision therefore has been made to perform mapping on three following standards that are essential to improve software testing practices in the medical device industry: ISO/IEC/IEEE 29119-2 generic software testing best practice, IEC 62304 medical device SLCPs, and ISO 14971 risk management. As the proposed software testing framework consists of processes of international standards related to generic software testing and medical device software development, therefore relevant to address identified generic and medical device software testing challenges. Other standards identified by the review are suited to later integration as they provide non-software-testing V&V practices to further maintain the software quality. The software testing framework incorporates software testing best practice of ISO/IEC/IEEE 29119-2 and those clauses of IEC 62304 and ISO 14971 standards, which must be fulfilled to ensure testing compliance with medical device regulatory requirements. The framework consolidates clauses that come from multiple standards, defining their relationships. Consolidation based on defined relationships ensures that the implementation of the framework does not compromise compliance with of IEC 62304 and ISO 14971 in any way but extends medical device SLCPs with generic software testing of ISO/IEC/IEEE 29119-2.

This approach to the development is also considered as an example for the future development of the whole SQA best practice framework for medical device software. The standards review in Section 2.4 identified all standards suitable to address SQA challenges identified in Sections 2.2 and 2.3, therefore needed for this future development. For the development and evaluation of the software testing best practice framework, the appropriate research design has been selected. The research design, the philosophical background and the research design elements, are presented in Part 2.

Map of Thesis – Part 2



Chapter 3 Research Design

3.1 Introduction

This chapter presents an overview of the philosophical and methodological background of this study. Crotty (1998 pp. 2-3) describes a research design as consisting of four design elements:

A. ***Epistemology*** informing about the theory of knowledge (Feast and Melles 2010)
B. ***Theoretical perspective*** grounding the logic and criteria of the research process
C. The ***Methodology*** which governs the use of methods
D. ***Methods*** to obtain and analyse data

These research design elements are presented by Saunders et al. (2013) like layers of an onion. Figure 3 illustrates an adapted version of the research onion, which includes research design elements introduced by both, Saunders et al. and by Crotty. According to Crotty, the assumptions embedded in the primary element inform each subsequent element (1998 p.2).

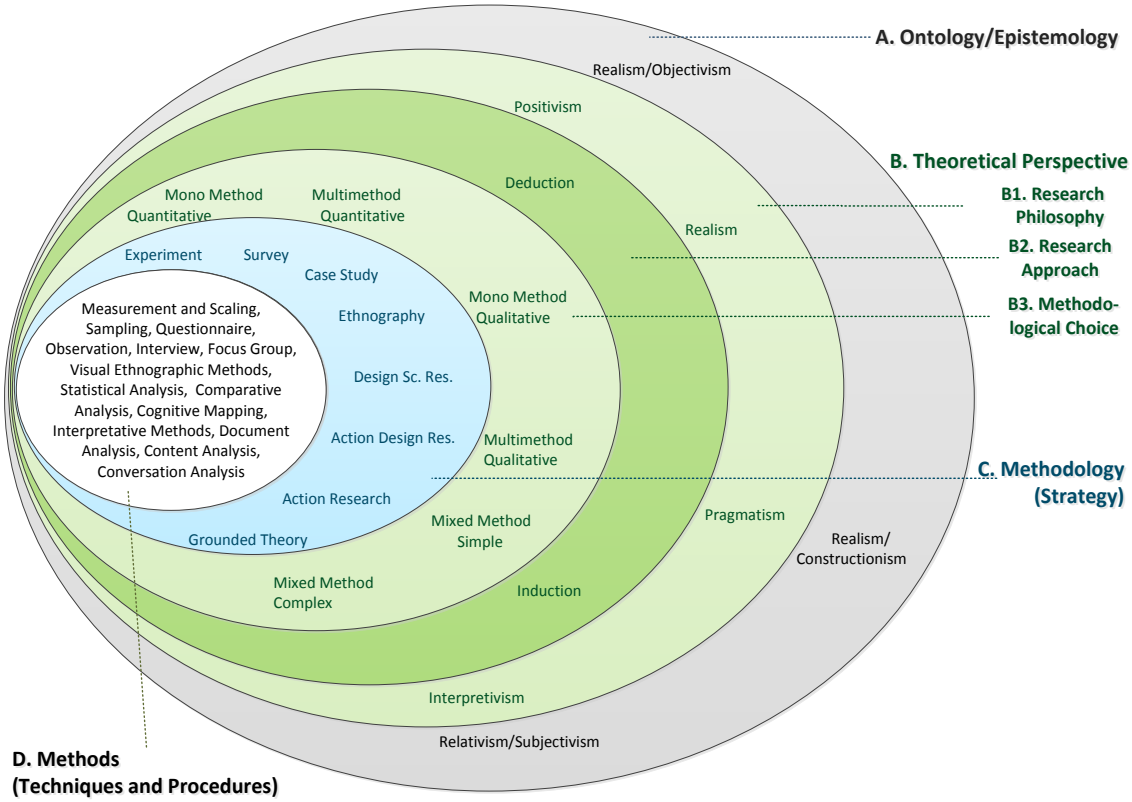


Figure 3 Research Onion adapted from Saunders et al. (2012, Crotty 1998)

The following section discusses these research design elements.

3.2 Philosophical and Methodological Background

3.2.1 Ontology and Epistemology

Ontology asks questions like *"What is existence and its nature?"*.

Epistemology asks questions like *"What do we know and how do we know it?"*.

Layer A. of the research onion in Figure 3 presents three ***Ontological/Epistemological*** philosophical views underlying various types of study. In this study, the philosophical background underpins acquiring and building knowledge about the software industry. The ontological/epistemological background will emerge from the following views based on the nature of this study:

Realism/ Objectivism - the reality or truth exists outside the mind, and the meaning of reality is independent of any consciousness, and it can be measured (Guba and Lincoln 1994).

Realism/ Constructionism - the reality or truth can exist outside the mind but needs to be interpreted, and reality can also be constructed. The engagement of the researcher in constructing enables learning as building knowledge structures (Papert and Harel 1991).

Relativism/ Subjectivism - there is no single reality or objective truth, and reality is what we perceive to be real. The knowledge about reality is purely a matter of perspective where each point of view has its truth (Thornhill et al. 2009 pp. 110-111).

3.2.2 Theoretical Perspective

Ontological and Epistemological philosophy underpins the researcher's view of the process of knowledge development in this study. The way a researcher understands a research question and the associated research design is known as a ***Theoretical Perspective***, ***Philosophical Stance*** or ***Research Paradigm*** (Saunders and Tosey 2013). The ***Theoretical Perspective*** consists of the ***Research Philosophy***, ***Research Approach*** and ***Methodological Choice***, grounding the logic and criteria of various studies. These components of the theoretical perspective are described in the following subsections.

3.2.2.1 Research Philosophy

The main research philosophies underlying the logic of identifying and addressing the research problem in this study are presented in Figure 3 Layer B1. of the research onion.

Positivism represents the researcher's view that the world exists independently of humans, and there is a single truth or reality which can be observed and investigated (Thornhill et al. 2009 pp. 113-114).

Realism Saunders at al. (2009) describes two types of realism: ***Direct Realism*** and ***Critical Realism***. Within ***Direct Realism***, we can perceive the world through our senses, while in ***Critical Realism*** our perception of the world is influenced by our senses and thus the world is not experienced directly.

Interpretivism This research philosophy views knowledge as the reality dependent on human practices and interaction with the world (Crotty 1998). The researcher's mind is not neutral in constructing knowledge, therefore, there is no one truth but more than one explanation of what is happening is possible (Oates 2005 p. 292).

Pragmatism is not committed to any philosophy or reality. The research problem and related research question are more important than the research method, and both, ***Positivist*** and ***Interpretive*** research philosophies can be applied to understand the truth (Thornhill et al. 2009 p. 109).

3.2.2.2 Research Approach

The ***Research Approach*** in this study is determined by the Research Questions set out in Section 1.5 and the way the Research Objectives presented in Section 1.6 are conducted to answer the Research Questions. Layer B2. of the research onion in Figure 3 consists of two approaches, ***Inductive*** and ***Deductive***.

The ***Inductive*** approach is known as a bottom-up approach which moves from specific to general. The researcher collects qualitative data, analysis of which provides an understanding of the research context. Based on this understanding, the researcher develops an initial hypothesis and then theory.

The ***Deductive*** approach is known as a top-down approach in which a hypothesis and a theory are developed. Based on the research strategy performed to test the hypotheses; the hypotheses are confirmed or rejected. Depending on the findings, the theory is then modified if the hypotheses are rejected.

3.2.2.3 Methodological Choice

The ***Methodological Choice*** determines the data collection technique and the type of data collected that are appropriate to answer the Research Question in this study. The choice of the data type and data collection technique is made from methodological choices represented by Layer B3. of the research onion in Figure 3.

Quantitative research gathers quantitative data from a high number of respondents with the emphasis on measurement or classification of variables (Dawson 2009, p. 15).

Qualitative research explores attitudes, behaviours and experiences and emphasises meanings of words rather than frequencies and distributions of numbers when collecting and analysing data (Dawson 2009, pp. 14-15).

Mono-Method uses a single data collection technique which can be either quantitative or qualitative and corresponding analysis procedures.

Multi-Method uses more than one data collection techniques, which can be either all quantitative or all qualitative techniques.

The ***Mixed-Methods*** approach uses both quantitative and qualitative collection techniques.

3.2.3 Methodology

The ***Methodology*** governs the use of the methods needed to accomplish the Research Objectives to answer the Research Question. The ***Methodology*** in this study is determined by the main Research Objective, the need to develop an innovative artefact, which is the software testing best practice framework for medical device software. The artefact is something constructed by humans, as opposed to something that occurs naturally (Simon 1996). Layer C. of the research onion in Figure 3 represents ***Methodologies*** that describe the plan of actions to drive various studies.

The ***Experiment*** aims to study causal links between variables and answer exploratory 'how' and explanatory 'why' questions (Thornhill et al. 2009 p. 142). Experiments are often conducted in laboratories rather than in the industry organisations and have great control over aspects of the research process.

Surveys enable the collection of a large amount of quantitative data from a sizeable population and answers 'who', 'what', 'where', and 'how much' questions (de la Vara et al. 2014). Surveys also explain relationships between variables and possible reasons for these relationships. The generally used data collecting techniques are questionnaires, structured observations and structured interviews.

A ***Case Study*** aims to gain a rich understanding of the real-life context of the study and the phenomenon of being studied (Tellis 1997). A single case can be an industry organisation. The data collecting techniques include interviews, observations, document analysis and questionnaires. Case study answers 'why', 'what' and 'how' questions.

Ethnography is rooted in the inductive approach and aims to describe and explain the social world (Thornhill et al. 2009 pp. 149-150). Ethnography aims to research the

phenomenon within the context in which it occurs and often involves extended participant observation as a data collecting technique.

Design Science Research aims to introduce an innovative artefact and the process for building this artefact within an application domain (Simon 1996). Artefacts are rarely developed enough to be used in practice (Fernández et al. 2012, Hevner 2007). Rather, they introduce innovative practices that can be fully accomplished and implemented in the future. In the beginning and throughout the research process, the position of the researcher is dominant.

Action Design Research generates prescriptive design knowledge through the building and evaluating ensemble information technology artefacts in an organisational setting (Sein et al. 2011). Therefore, this strategy involves close collaboration with industry organisation and collaborative artefact, which is shaped by both designer and user.

Action Research is a highly context-dependent methodology yet addresses a specific industry problem (Iivari and Venable 2009). Therefore, action research involves collaboration between the researcher and industry practitioner and the role of the practitioner is dominant at the beginning of the research (Järvinen 2012).

Grounded Theory is usually used in social sciences for observing a group to collect and analyse data and construct a theory which is grounded in the data (Dawson 2009, p. 19). It is an inductive approach where data is collected by techniques such as focus groups, literature review or interviews.

3.2.4 Data Collection Methods

Layer B3. discussed quantitative and qualitative research methodological choices such as Mono, Multi and Mixed-Methods, while Layer D. of the research onion in Figure 3 is concerned with the data generated by this study and data collection methods.

Quantitative Data is numerical or data that could be quantified (Thornhill et al. 2009 p. 151). Examples of appropriate data collecting methods include surveys or questionnaires with close-ended questions and scoring options.

Qualitative Data is non-numerical, and examples of collection methods are an individual interview or focus group discussion (Thornhill et al. 2009 p. 151).

An overview of the philosophical and methodological background presented in Section 3.2 is followed in Section 3.3 by a description of and the reasoning for the choices of the research design elements needed for the development and evaluation of the software testing best practice framework for medical device software.

3.3 Research Design of this Study

Figure 4 presents the research design elements needed to develop and evaluate the software testing best practice framework for medical device software, based on the basic research design elements presented in Figure 3.

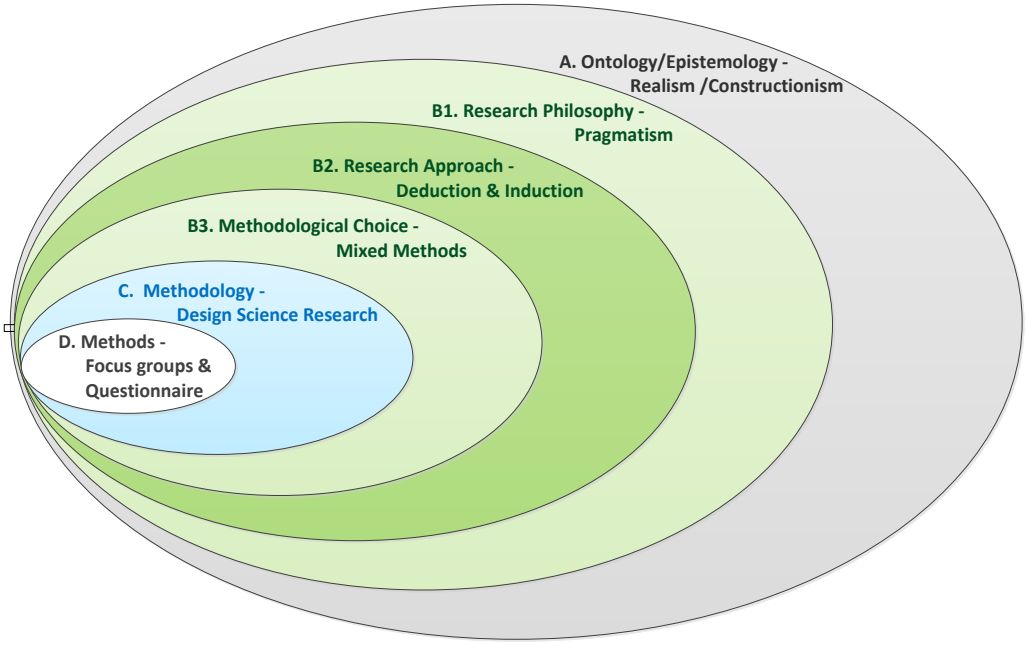


Figure 4 Research Choices adapted from Saunders at al. (2012, Crotty 1998)

3.3.1 Ontological/Epistemological View

This study adopted ***Realist Ontology*** and ***Constructionist Epistemology*** as the philosophical background for the research design. Chapter 1 of this thesis specifies that this study focuses on reality in the software development industry. The software industry and related problems exist regardless of a personal view. Their existence is supported in Chapter 2 – literature review by evidence of software quality and safety issues. Studying a real problem that exists regardless of our personal view is underpinned by ***Realist Ontology*** which supports studying what really exists regardless of perception.

Section 2.5 identified the research problem in this study as a lack of consolidated information on the implementation of software testing best practice in the medical device domain and proposed the solution as the development of a software testing best practice framework for medical device software. The development based on mapping generic software testing standard to medical device software development standards and identifying their relationships, represents the construction of a new reality. Reflection on the development by validating this framework with industry organisations enables learning and provides knowledge. This approach is underpinned by ***Constructionist Epistemology*** according to which a new reality can be created and interpreted.

3.3.2 Theoretical Perspective of this Study

The ***Theoretical Perspective*** in this study is determined by adopting the ***Philosophy of Pragmatism***, ***Inductive*** and ***Deductive Approaches*** and ***Mixed Methods***.

Prior to this study, research questions (Section 1.4) were formulated. The determination of the research objectives (Section 1.5) and the ***Theoretical Perspective*** underlying these objectives were driven by these questions. Based on the identified software testing challenges (Sections 2.2 & 2.3) and standards contributing to addressing these challenges (Section 2.4), the software testing best practice framework was developed. The framework was then validated by industry organisations to test the impact of this framework on implementing software testing best practice for medical device software. The problem-solving approach emphasized the practical implications of the study rather than the research method. The need for answers to research questions and the emphasis on solving an industry problem without being constrained by a rigorous methodology are aligned with the ***Pragmatism*** perspective that is not committed to any research philosophy and allows any research approach to be used to answer research questions.

The research perspective of ***Pragmatism*** supports the use of both ***Inductive*** and ***Deductive Approaches***. The literature and standard review conducted as a part of this study was focused towards an ***Inductive Approach***. Software testing challenges in the medical device domain and the use of standards contributing to addressing these challenges were collected through this bottom-up approach. Analysis of the collected information provided the basis for determining the research problem. The development and validation of a software testing best practice framework were focused towards an ***Deductive Approach***. With this top-down approach the hypothesis was articulated that: *"An approach to consolidate generic software testing best practice with related requirements of harmonised medical device software standards has an impact in addressing identified software testing challenges"*. This framework was validated to confirm or reject the hypothesis.

Mapping standards' clauses to develop a framework, learning from the development approach to build knowledge structures, and evaluating the framework's impact to improve testing of medical device software involve mainly the collection of ***Qualitative Data***. To analyse obtained ***Qualitative Data***, additional ***Quantitative Data*** was collected through a questionnaire scoring the impact of the framework. ***Mixed Methods*** were therefore proposed to collect both ***Qualitative*** and ***Quantitative Data*** in this research (Creswell 2016).

3.3.3 Methodology

This study adopted the ***Design Science Research (DSR)*** methodology. The aim of this study was to develop an innovative artefact – a software testing best practice framework for answering research questions. ***Methodologies*** such as ***Action Research***, ***Action Design Research*** and ***Design Science Research*** are oriented on the development of an innovative artefact. ***Action Research*** and ***Action Design Research*** aim to address a specific problem in an organisational setting and involve collaboration with industry practitioner in a dominant position at certain stages of the study.

Rather than solving a specific problem within the organisation, this study aimed to address the problem category of software testing in the medical device software domain. The goal of this study was to introduce an innovative practice of medical device software testing that can be used and improved in the future. Since ***DSR*** is oriented on developing an innovative artefact within an application domain, with the researcher's dominant position in the entire study, ***DSR*** was considered the appropriate methodology for this study. Hevner et al. provided ***DSR*** guidelines, presented in Table 1.

Table 1 Design-Science Research Guidelines (Hevner et al. 2004)

Number	Guideline	Description
1	Design as an Artefact	Design-science research must produce a viable artefact in the form of a construct, a model, a method, or an instantiation.
2	Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
3	Design Evaluation	The utility, quality, and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods.
4	Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artefact, design foundations, and design methodologies.
5	Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artefact.
6	Design as a Search Process	The search for an effective artefact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
7	Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

In addition to these guidelines, Peffers et al. (2007) provide activities for ***DSR*** methodology as follows: problem identification and motivation; define the objectives for a solution; design and development; and demonstration, evaluation and communication.

According to Hevner (2007), ***DSR*** is based on three cycles: ***Relevance Cycle***; ***Rigour Cycle***; and ***Design Cycle***. Figure 5 presents a direct adaptation of the original figure from Hevner for the development of the software testing best practice framework for medical device software in this study (2007).

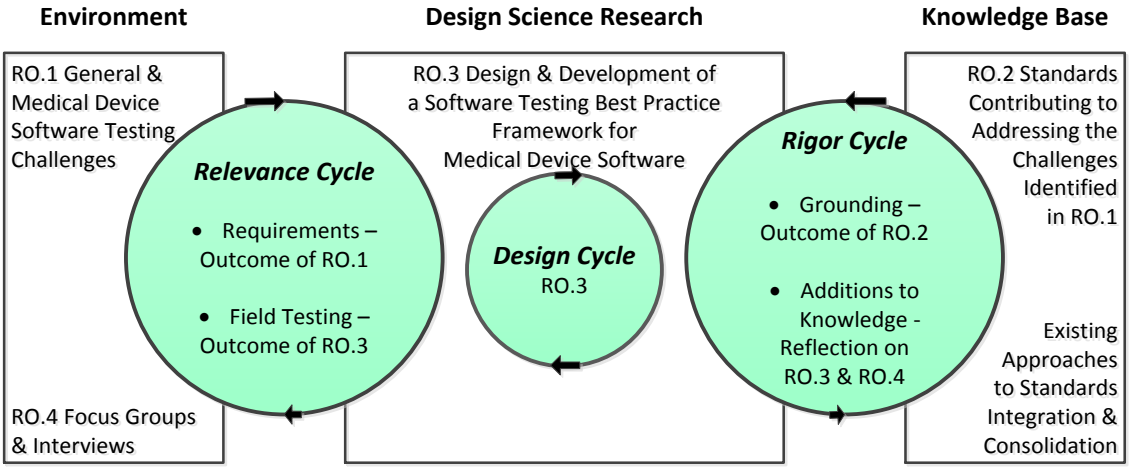


Figure 5 DSR Cycles of Development of a Framework adapted from Hevner (2007)

To answer RSQ.1, the literature review examined the software testing challenges that need to be addressed in the medical device domain (Section 2.3). The ***Relevance Cycle*** on the left side of the diagram provided the requirement to address the identified challenges into the ***Design Cycle***. To answer RSQ.2, international standards were reviewed (Section 2.4) to identify existing standards contributing to addressing the identified challenges. The use of existing knowledge base to address problems is a part of DSR. The ***Rigour Cycle*** on the right side of the diagram provided identified standards and knowledge about existing approaches to their integration and consolidation into the ***Design Cycle***. To answer RSQ.3, in the ***Design Cycle***, the development of the framework took place (Chapter 4). Another task of the ***Relevance Cycle*** returned the framework to industry for validation (Chapter 5) to answer RSQ.4. The ***Rigour Cycle*** provided the lesson learned from developing and validating the framework as a basis for the future development of a SQA best practice framework for medical device software (Section 6.4), and provided additions to knowledge by publishing in Joint Proceedings of 11th Systems Testing and Validation Workshop and 3rd International Workshop on User Interface Test Automation. From the above description, ***DSR*** based on three cycles has been considered as an appropriate methodology for developing an innovative framework to answer research question. Therefore, three cycles of ***DSR*** and related terminology are used in this study as a research strategy to design and develop a software testing best practice framework for medical device software.

3.3.4 Data Collection Methods

This study involved obtaining mainly qualitative data in order to evaluate the framework by industry organisations. The data collected was related to the overall structure and impact of the framework in improving medical device software testing practices. To obtain rich data information, it was considered that validation participants should be allowed to express a wide spectrum of their views. Using a ***Focus Group*** enables a broad perspective and a collection of unique and rich qualitative data on the research subject. This data is obtained as feedback from respondents discussing their opinions and, consequently, inspiring or influencing each other. Therefore, ***Focus Group*** was deemed to be an appropriate data collection technique and was used in this study to gain qualitative data from medical device software development organisation and generic software testing organisation.

While qualitative data collection allows a broad feedback perspective and is therefore considered to be the main source of validation, quantitative data makes it possible to analyse the data obtained. This allows qualitative data from focus groups to be broadened by assessing the extent to which participants perceive the impact of the framework in improving medical device software testing practice. Using ***Questionnaire*** enables to obtain quantitative data and was therefore considered an appropriate data collection technique in this study. A ***Questionnaire*** was used to evaluate close-ended questions with scoring from 1 to 5, which indicated the level of the framework's efficiency.

To obtain rich qualitative feedback and measure the data obtained, both qualitative and quantitative data were collected to assess the perceived impact of the software testing best practice framework for medical device software. The feedback provided information on the overall structure of the framework and its impact in improving industry medical device software testing practices.

3.4 Summary

Realism/Constructionism supports the study of reality and the creation of a new reality. Therefore, ***Realism/Constructionism*** has been considered as an appropriate Ontological/Epistemological philosophical background for studying software testing challenges, developing a framework for testing medical device software and learning from the development and validation of the framework.

The emphasis on the research practical implications and the problem-solving approach in this study make addressing research questions more important than the research method. The development of the software testing best practice framework for medical device software was determined by the research questions which were articulated based on the quality and safety issues of medical device software. Since the ***Pragmatic*** research perspective is not committed to any research philosophy, and the research problem and related research question are more important than the research method, it was considered appropriate for this study.

In this study, a bottom-up approach was taken, using literature and standards review to identify the research problem. While software testing challenges in the medical device domain were identified by literature review, the standards contributing to addressing these challenges were collected through standards review. Analysis of the collected information provided the basis for determining the research problem as lack of consolidated information on the implementation of software testing best practice for medical device software.

A top-down approach was used to address the research problem by formulating and testing a hypothesis that: *"An approach to consolidate generic software testing best practice with related requirements of harmonised medical device software standards has an impact in addressing identified software testing challenges"*. Therefore, both the ***Inductive*** bottom-up approach to the literature and standards review and the ***Deductive*** top-down approach in designing and developing the software testing best practice framework for medical device software were considered appropriate for this study.

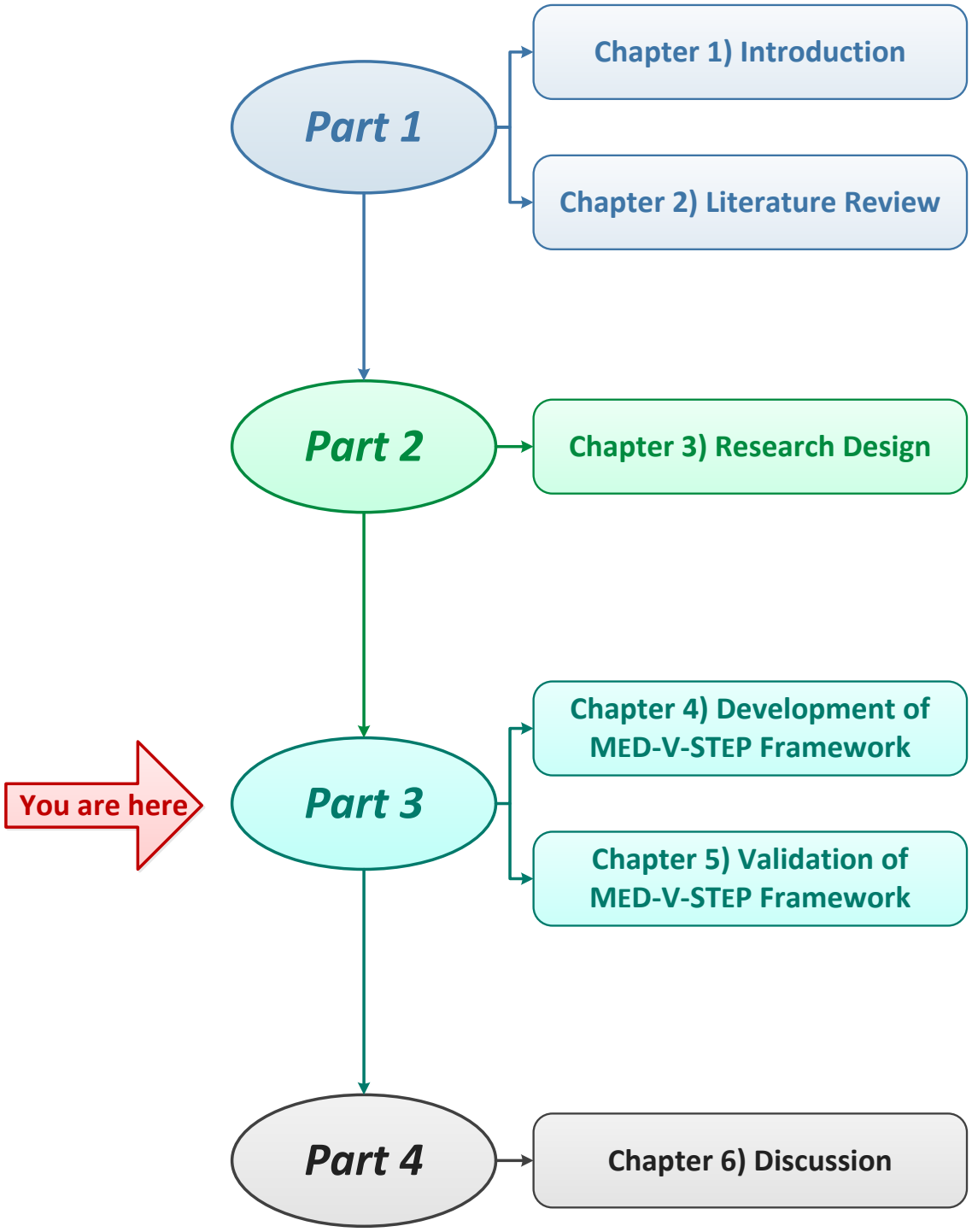
Qualitative data on the challenges of software testing, the development of the framework as a solution and the evaluation of the framework were considered appropriate during the review of the literature and the development and validation of the framework. However, in addition, ***Quantitative*** data was collected to give a measure to the ***Qualitative*** data

obtained. Therefore, a ***Mixed-Methods*** approach using both ***Qualitative*** and ***Quantitative*** data was used in this study.

The aim of this study was to solve the identified problem of software testing in the medical device domain by introducing an innovative software testing best practice framework for medical device software. Therefore, ***DSR***, which supports the development of an innovative artefact to address the problem category, was considered appropriate and used in this study.

The aim of validation was to primarily obtain qualitative feedback from industry. Using the ***Focus Group*** method enabled rich qualitative data collection, and was therefore valuable in this study. In order to address the need to provide addition to knowledge base and assess the extent to which validation participants perceive the impact of the framework in improving medical device software testing practices, the qualitative feedback obtained was extended to include quantitatitve data collected by ***Questionnaires***.

Map of Thesis – Part 3



Chapter 4 Development of a Software Testing Best Practice Framework

4.1 Introduction

This chapter describes the approach taken to design and develop an innovative software testing best practice framework for medical device software in accordance with the research design in Section 3.3. When developing the framework as outlined in RO.3, the international standards identified in RO.2 have been used to address the generic and medical software testing challenges identified in RO.1. This framework addresses these challenges using generic software testing best practice to extend testing clauses in the medical device software standards required for regulatory compliance.

The use of multiple standards is difficult due to the need to identify and integrate software test clauses that are distributed in these standards, and to consolidate them by identifying their relationships to each other. Based on the output of RO.2, the research problem in this study was identified as the lack of consolidated information in the form of a single standard for testing medical device software. The approach used in this thesis to address the research problem was to adapt the most recent best practice in generic software testing to the medical device software standards. According to *ISO/IEC/IEEE 29119-1*, testing is based on regulations and standards applicable to the specific industry, but this standard does not inform how to base its processes on medical device standards.

As stated in Section 2.4 of the standards review, ISO/IEC 12207 standard which provides generic best practice software life-cycle processes was used as the basis for the development of IEC 62304. Therefore, most of the clauses of IEC 62304 are mapped to the corresponding clauses of ISO / IEC 12207, as published in the mapping table attached to IEC 62304. The clauses of ISO/IEC/IEEE 29119-2 software testing best practice are aligned with clauses of generic software life-cycle processes of ISO/IEC 12207, and this alignment is published in the mapping table attached to ISO/IEC/IEEE 29119-2. Since ISO/IEC 12207 has been mapped to IEC 62304 as well as ISO/IEC/IEEE 29119-2, it can therefore be used to align IEC 62304 with ISO/IEC/IEEE 29119-2. This alignment was achieved by mapping ISO/IEC/IEEE 29119-2 software test clauses to the relevant IEC 62304 life-cycle clauses via the existing mappings of these standards to ISO/IEC 12207. The software test clauses with related life-cycle and risk management clauses were integrated into a framework and consolidated by defining their logical dependencies, as presented in the following section.

The approach to the development of this testing framework was based on the existing approaches presented in Section 2.6. The proposed software testing framework aims to address the identified software testing challenges and help organisations implement sets of necessary test activities without having to identify, collect and integrate them from multiple standards. The defined logical dependencies aim to help organisations maintain relationships between activities that come from various standards without having to define them, so that the activities are performed in sequence where the generation or change of the initial activity affects subsequent activities. The framework also prevents the implementation of the multiple standards clauses twice or separately.

4.2 Framework Design

The software testing framework is intended for software development organisations already compliant with IEC 62304 requirements that have not adopted software testing best practice. The scope of the designed and developed **Me**dical **D**evice **V**erification **S**oftware **Te**st **P**rocesses (MED-V-STEP) framework is presented in Figure 6. IEC 62304 requires verification to be performed for each development stage. Therefore, the framework design focuses on the area of software testing within verification and comprises of generic software test processes within medical device software verification. This was achieved by adapting the generic software test processes of ISO/IEC/IEEE 29119-2 to the IEC 62304 medical device software life-cycle processes and ISO 14971 risk management process as described below.

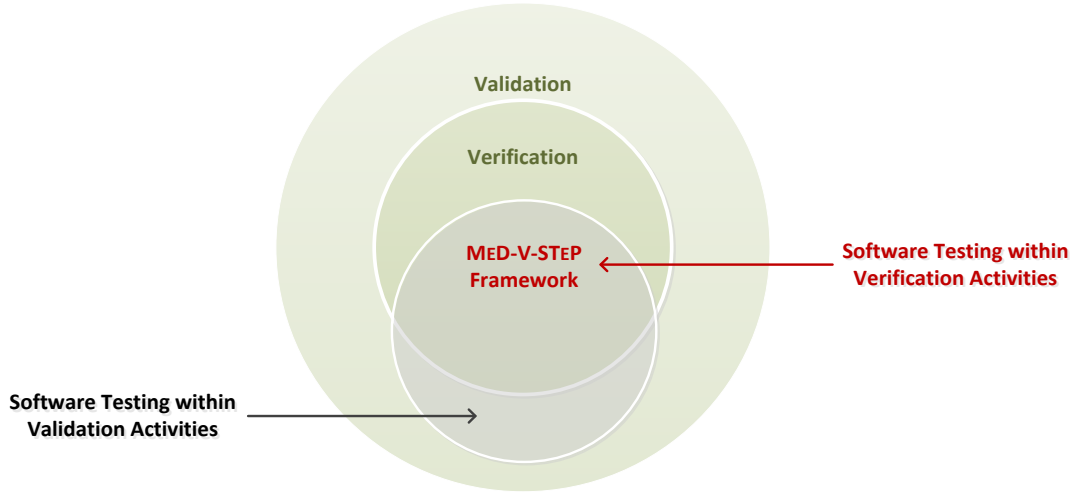


Figure 6 MED-V-STEP within SQA – adapted from Vogel (2011 p. 77)

For manufacturers who have implemented IEC 62304 software life-cycle processes, the implementation of software testing framework containing all IEC 62304 clauses would be a duplication of the IEC 62304 clauses that have been already implemented. The identified clauses in IEC 62304 that relate to software testing and that could benefit from implementing software testing best practice in the context of the implementation of the overall ISO/IEC/IEEE 29119-2 test processes. Therefore, the MED-V-STEP framework does not include all IEC 62304 clauses, but only those identified as related to ISO/IEC/IEEE 29119-2 clauses, providing information where the relationship between test and development activities takes place.

ISO/IEC/IEEE 29119-2 requires, that all testing decisions made, are informed by the knowledge of the current risk situation (Reid 2012). According to this standard, risk management standards can be applied with ISO/IEC/IEEE 29119-2 to determine the risks

associated with the use of the software system (2013a p. 24). IEC 62304 requires risk management to be performed and refers to the application of risk management according to the requirements of ISO 14971 *Medical devices – Application of risk management to medical devices* (ISO 2012). The ISO 14971 standard which is indispensable for the application of risk management within IEC 62304 also can be applied to the risk-based software testing approach of ISO/IEC/IEEE 29119-2. Therefore, the development of the MED-V-STEP framework also includes those ISO 14971 clauses that were identified in this study as related to software testing. Figure 7 shows that the MED-V-STEP framework includes those ISO 14971 risk management clauses that are referred to in IEC 62304 development clauses related to ISO/IEC/IEEE 29119-2 software test clauses.

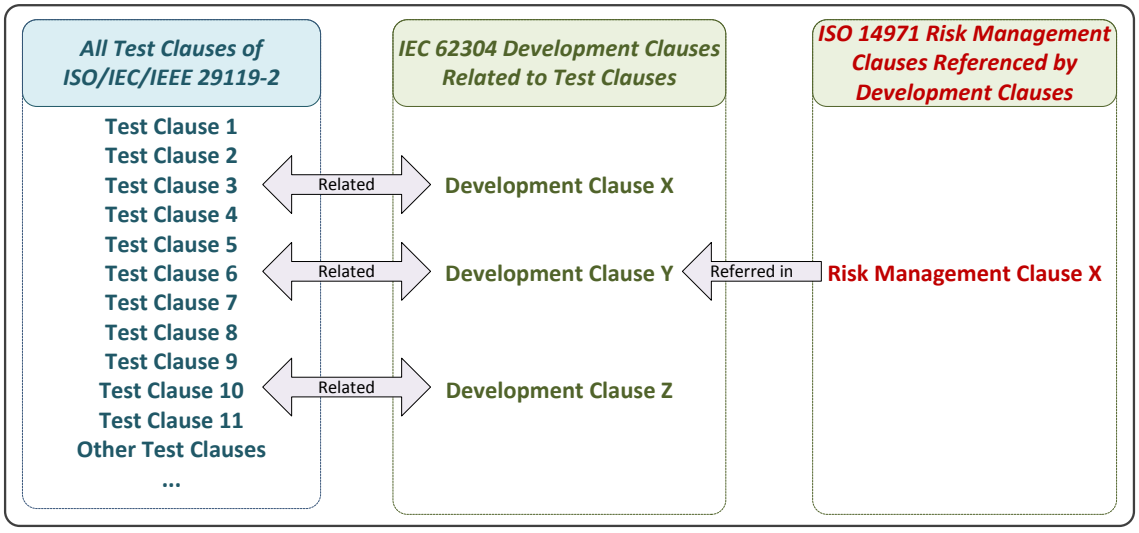


Figure 7 MED-V-STEP Framework Design Including Related Clauses of IEC 62304 and ISO 14971

The following section describes the initial high-level mapping of ISO/IEC/IEEE 29119-2 and IEC 62304 to assess the potential for consolidation of their processes.

4.3 High-Level Mapping of ISO/IEC/IEEE 29119-2 & IEC 62304

The mapping of standards' processes, activities and tasks is the approach taken in this study to the development of the MED-V-STEP framework. Therefore, a high-level mapping was initially performed to assess the feasibility of detailed mapping. The high-level mapping process was conducted, and Figure 8 presents the output of this mapping.

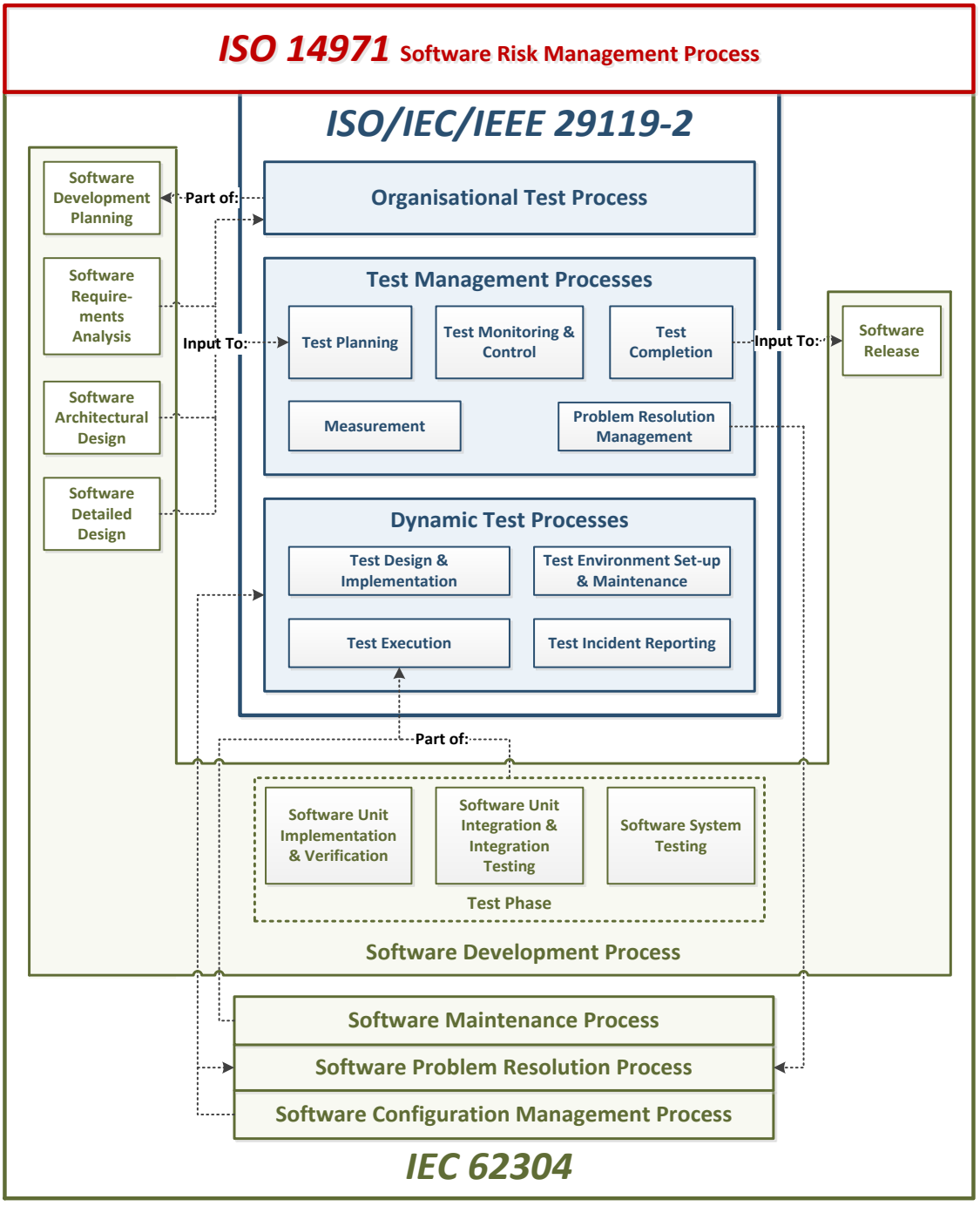


Figure 8 High-Level Mapping of ISO/IEC/IEEE 29119-2 & IEC 62304

The high-level mapping examined the relationships at the level of test processes of ISO/IEC/IEEE 29119-2 and development activities of IEC 62304. The identified relationships presented in Figure 8 are the following:

- organisational test process is a part of software development planning activity
- software requirements analysis, software architectural design and detailed design activities are inputs to the organisational test process and test planning process - the first stage of test management processes
- the test phase of the software development process and software maintenance process were recognised as a part of the execution test process, a stage of dynamic test processes
- software configuration management process is an input to software resolution process and dynamic test processes
- problem resolution management process is a part of software problem resolution process
- test completion process is an input to software release activity

These relationships between processes and activities of the mapped standards identified by high-level mapping presented in Figure 8, provided the basis for a low-level detailed mapping and consolidation of related clauses as described in Section 4.4. High-level mapping and approach to the development of a software testing best practice framework for medical device software were presented in Ireland to a Software Testing Organisation and at an international 11th Systems Testing and Validation Workshop. Since the workshop was not closely related to the medical device domain, the purpose of the presentation was to provide feedback on the mapping of test activities to software development life-cycle, as this mapping could also apply to adapting generic software testing best practice to any other non-medical software development life-cycle. The feedback from both the manager from the Software Testing Organisation and the participants of the 11th Systems Testing and Validation Workshop confirmed the relevance of the high-level mapping and the approach to the development of the framework based on detailed mapping of the selected standards. The following section presents an approach to the consolidation of information distributed in ISO/IEC/IEEE 29119-2, IEC 62304, and ISO 14971.

4.4 Approach to Consolidating Standards' Clauses

The consolidation of the information from multiple standards does not mean only a simple collection of required clauses. The use of such a collection would not differ from simple compliance with international standards identified for the development of the testing framework. The difference between simple collection and the consolidation is that besides the identification of required clauses, consolidation also identifies existing relationships between these clauses that come from various standards. The relationship defines the related clauses and the nature of their relationship, that is the logical dependence. Defined logical dependencies have an impact on the implementation of software testing for medical device software by describing how the ISO/IEC/IEEE 29119-2 test clauses and the related clauses of IEC 62304 software life-cycle processes interact. Consolidated clauses enable organisations to implement ISO/IEC/IEEE 29119-2 test activities at the appropriate stage of the IEC 62304 software life-cycle processes and use the output of one activity as input to enhance the related activity. Therefore, the approach to the consolidation of multiple standards' clauses involves the identification of the related clauses as well as their logical dependencies, as described in the following section.

4.4.1 Logical Dependence of Standards' Clauses

According to ISO/IEC/IEEE 29119-1, testing is a set of interrelated or interacting activities that transform inputs into outputs (ISO/IEC/IEEE 2013a p. 13). The output of one related activity generates the input of another related activity. This standard defines the logical dependencies between related activities in terms of input and output as a basis for their efficient application. The adherence to a logical dependence requires that an activity generating an input to the related activity is performed prior to this related activity. For example, the *Identify & Analyse Risk* activity from ISO/IEC/IEEE 29119-2 must be conducted prior to *Identify Risk Mitigation Approach* activity, because the approach to mitigating risk can be identified based on risk analysis.

In IEC 62304, similar to ISO/IEC/IEEE 29119-2, the logical dependence determines the sequence in which the related clauses need to be performed. For example, a *Requirements Analysis* activity from IEC 62304 must be conducted prior to *Architectural Design* activity, because the output of requirements analysis generates the input needed for designing. In another example, the *Software Safety Classification* of the software system should be completed after the *Risk Analysis* process has established what harm could arise from the failure of the software system (IEC 2015 p. 43).

The rationale for this approach is included within the international standards selected for the development of the framework. These standards require the application of related clauses in adherence with the logical dependencies between these clauses. According to IEC 62304, the application of processes and activities that adhere to their logical dependencies is considered valuable in providing high-quality software development.

> "*Whenever any process output is created or changed, all related process outputs should be updated promptly to maintain their consistency with each other and to maintain all dependencies explicitly or implicitly required by this standard"* (IEC 2015 p. 44).

The need to define relationships between standards clauses also applies to the use of clauses from two or more international standards. However, defining relationships between multiple standards is more complex. To assist in defining these relationships, international standards include tables with high-level mapping to other standards. ISO/IEC/IEEE 29119-2 annexes contain tables defining how the clauses of ISO/IEC/IEEE 29119-2 are related to other standards, for example to ISO/IEC 12207 presented in Section 2.4.2 (ISO/IEC/IEEE 2013b pp. 42-52). These mappings explain how to make efficient use of related standards. Similarly, IEC 62304 annexes contain a number of tables, showing the relationships between IEC 62304 and other standards including ISO 14971. The table in Annexe C.3 contains the clauses of IEC 62304, which refer to the ISO 14971 risk management process and require its application (IEC 2015 p. 63).

The above description indicates that in order to ensure efficient implementation of ISO/IEC/IEEE 29119-2 test processes in conjunction with IEC 62304 and ISO 14971 standards, the MED-V-STEP framework should define the relationships between these standards. While these standards consist of interrelated clauses, the relationships between these standards were not defined prior to the development of this framework. Relationships and their logical dependencies were therefore defined for ISO/IEC/IEEE 29119-2, IEC 62304 and ISO 14971 in this study.

The MED-V-STEP framework maintains the structure, relationships and logical dependencies of test clauses under ISO/IEC/IEEE 29119-2 as basis for their efficient implementation, where the output of one test activity generates the input of another test activity. At the same time, the framework maintains relationships and logical dependencies defined in this study between ISO/IEC/IEEE 29119-2, IEC 62304 and ISO 14971, enabling the implementation of sets of ISO/IEC/IEEE 29119-2 test clauses at the

appropriate IEC 62304 life-cycle stage, where the output of test activity generates the input of development activity, or the output of development activity generates the input of test activity. If software test, development and risk management processes were applied in isolation without maintaining their relationships, there would be a risk that the related activities from these processes would not interact with each other. Defining logical dependencies of related clauses is described in detail in Section 4.5 and examples are given in Section 4.6.

4.4.2 Summary and Conclusion

Section 4.4 presented the approach taken in this study to consolidate generic software testing best practice of ISO/IEC/IEEE 29119-2 with testing related clauses from IEC 62304 and ISO 14971. The consolidation approach involves identification of existing relationships between clauses within these standards. The first step in consolidation defines the related clauses. The next step in consolidation is to define logical dependencies enabling to determine how related clauses interact and affect each other.

The justification for this approach is contained in ISO / IEC / IEEE 29119-2 as well as in IEC 62304. Both standards emphasize the importance of maintaining the logical dependencies of interrelated clauses for the high efficiency implementation of their processes. They also emphasize the importance of maintaining logical dependencies of related clauses from different standards for high efficiency of their joint implementation. To this end, the annexes of these standards contain high-level mapping tables for two different standards, such as the annex in ISO/IEC/IEEE 29119-2 which maps its clauses to ISO/IEC 12207, or the annex in IEC 62304 that maps its clauses to ISO 14971.

However, for the efficient combined implementation of ISO/IEC/IEEE 29119-2 with IEC 62304 and ISO 14971, there are no tables in the annexes that map their processes. Therefore, in this study, an approach has been taken to map these standards to identify the related clauses of these standards and define their logical dependencies. And on this basis to develop a framework providing information on the efficient combined implementation of ISO/IEC/IEEE 29119-2 software testing best practice in relation to relevant IEC 62304 and ISO 14971 clauses.

4.5 Development of MED-V-STEP Framework

The initial high-level mapping of ISO/IEC/IEEE 29119-2 and IEC 62304 has demonstrated the potential to consolidate information on medical device software testing. As presented in Section 4.3, the validity of high-level mapping was confirmed by Software Testing Organisation in Ireland and international 11th Systems Testing and Validation Workshop in Germany.

Following the high-level mapping, a detailed low-level mapping of these standards was conducted to identify related activities and tasks. Standard clauses are at the level of processes, activities and tasks. The highest level of standards' clauses represents processes which consist of activities and these, in turn, consist of tasks. The standards structure with clauses at various levels enables the definition of relationships at these levels. Tasks represent the lowest level of the standard's clauses and provide a most detailed description of what needs to be done to fulfil the standards' requirements. Efficient joint implementation of ISO/IEC/IEEE 29119-2 and IEC 62304 needs precise definition of their relationships at the lowest possible level. Precise information on related clauses and the nature of relationships enables to deliver the output of one activity as an input to enhance another activity, thereby improving the efficiency of the process.

The approach taken in this study to defining relationships of standards' clauses makes use of the Table B.2 in Annex B of ISO/IEC/IEEE 29119-2 and Table C.5 in Annex C.6 of IEC 62304. These tables show how the clauses of ISO/IEC/IEEE 29119-2 and IEC 62304 are related to ISO/IEC 12207 clauses. There is no table in annexes showing the relationships of ISO/IEC/IEEE 29119-2 with IEC 62304 directly. However, the presented in Figure 9 structure of tables with defined relationships to ISO/IEC 12207 was used in this study to perform an indirect mapping between these two standards and derive the existing relationships between the ISO/IEC/IEEE 29119-2 and IEC 62304 clauses.

Table B.2 in Annex B of ISO/IEC/IEEE 29119-2

ISO/IEC/IEEE 29119-2	ISO/IEC 12207
Test Clause 1	Clause 2
Test Clause 2	Clause 4
Test Clause 3	Clause 5
Test Clause 4	Clause 7

Table C.5 in Annex C.6 of IEC 62304

ISO/IEC 12207	IEC 62304
Clause 1	Development Clause 1
Clause 2	Development Clause 2
Clause 3	Development Clause 3
Clause 4	Development Clause 4

Figure 9 Mapping ISO/IEC/IEEE 29119-2 to ISO/IEC 12207 and ISO/IEC 12207 to IEC 62304

The following sections describe the use of the tables in annexes ISO/IEC/IEEE 29119-2 and IEC 62304 to perform mapping in this study.

4.5.1 Use of Table B.2 in Annex B of ISO/IEC/IEEE 29119-2

ISO/IEC/IEEE 29119-2 contains table B.2 in Annex B which provides a high-level explanation on how this software testing standard maps to the testing related clauses of ISO/IEC 12207. ISO/IEC 12207 defines requirements for generic, non-domain-specific software life-cycle processes (2008). This mapping is intended for the users of ISO/IEC 12207 to explain how to make use of ISO/IEC/IEEE 29119-2 (2013b p. 42). A sample of this mapping contained in ISO/IEC/IEEE 29119-2 is presented in Table 2.

Table 2 Sample of Table B.2 mapping ISO/IEC/IEEE 29119-2 to ISO/IEC 12207 (2013b p. 42)

ISO/IEC 12207 Clause	ISO/IEC/IEEE 29119-2 Clause	Mapping Explanation
6.1.2.3 Activities and Tasks 6.1.2.3.4 Contract Execution Task 6.1.2.3.4.15	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4 Activities and Tasks 7.3.4.4 Report (TMC4)	*This is supported by clause 7.3.4.4 Report (TMC4), which provides an ability to report testing progress and communicate new risks to stakeholders.*
	7 Test Management Processes 7.4 Test Completion Process 7.4.4 Activities and Tasks 7.4.4.4 Report Test Completion (TC4)	*This is also supported by clause 7.4.4.4 Report Test Completion (TC4), which provides the ability to report the outcomes of testing to stakeholders.*
6.2.1 Life-Cycle Model Management Process		
6.2.1.3 Activities and Tasks 6.2.1.3.2 Process Assessment Task 6.2.1.3.2.2	6 Organisational Test Process	*This is supported by clause 6 Organisational Test Process, which supports the periodic review of organisational test documentation, which can include the definition of processes for supporting testing.*
6.3.1 Project Planning Process		
6.3.1.3 Activities and Tasks 6.3.1.3.2 Project Planning Task 6.3.1.3.2.1	7 Test Management Processes 7.2 Test Planning Process	*The planning of testing is supported by clause 7.2 Test Planning Process, which provides generic processes for planning any phase or type of testing.*

Besides mapping of related clauses, Table B.2 provides an explanation of their relationship. The approach of providing explanation was used in the development of the MED-V-STEP framework and is referred to as a logical dependence of related clauses.

The test clauses presented in the middle column in this table define test processes and activities. There are three process groups consisting of eight test processes contained in ISO/IEC/IEEE 29119-2 divided into thirty-three activities. Each activity consists of several tasks, usually between two to seven. Fourteen out of thirty-three test activities of ISO/IEC/IEEE 29119-2 are mapped to software development tasks of ISO/IEC 12207.

ISO/IEC 12207 tasks, in turn, are mapped to IEC 62304 activities and tasks, as described in the next section. Due to a high detail at the level of activities and tasks, these related clauses provided a sound basis and were used for mapping in this study and consolidation of ISO/IEC/IEEE 29119-2 test clauses with IEC 62304 development clauses.

Thirteen dynamic test activities are mapped at the level of a process group – *Dynamic Test Processes* of ISO/IEC/IEEE 29119-2. For the development of the framework, the low detail at the level of a process group means that although the activities and tasks of this process group were also included in the framework, they are not directly related to any development clauses of IEC 62304. They provide information on test activities not contained in IEC 62304 and therefore, while not mapped to development activities or tasks, they enhance IEC 62304 by providing guidance on software testing best practice.

4.5.2 Use of Table C.5 in Annex C.6 of IEC 62304

IEC 62304 was developed based on ISO/IEC 12207 (IEC 2008 p. 73) which defines requirements for generic software life-cycle processes, and was tailored to the need of the medical device domain. Table C.5 in Appendix C.6 of IEC 62304 displays the relationship between IEC 62304 and ISO/IEC 12207, which is presented in Table 3.

Table 3 Sample of Table C.5 mapping IEC 62304 to ISO/IEC 12207 (2015 p. 74)

IEC 62304		ISO/IEC 12207	
Activity	Task	Processes	Activity/Task
5 Software Development Process			
5.1 Software Development Planning	5.1.1 Software Development Plan	7.1.1 Software Implementation 6.3.1 Project Planning	7.1.1.3.1 Software Implementation Strategy 7.1.1.3.1.1 7.1.1.3.1.3 7.1.1.3.1.4 6.3.1.3.2 Project Planning 6.3.1.3.2.1
	5.1.2 Keep Software Development Plan Updated	***6.3.2 Project Assessment and Control***	***6.3.2.3.2 Project Control 6.3.2.3.2.1***
	5.1.3 Software Development Plan Reference to System Design and Development	6.4.3 System Architectural Design 6.4.5 System Integration 7.2.5 Software Validation	6.4.3.3.1 Establishing Architecture 6.4.3.3.1.1 6.4.5.3.1 Integration 6.4.5.3.1.1 7.2.5.3.1 Process Implementation 7.2.5.3.1.4

The corresponding clauses of these standards are mapped and located in the same row of the table. For example, on the left side of the table, task *5.1.2 Keep Software Development*

Plan Updated of activity *5.1 Software Development Planning* of IEC 62304 is mapped on the right side to the task *6.3.2.3.2.1* of activity *6.3.2.3.2. Project Control* and process *6.3.2 Project Assessment and Control* of ISO/IEC 12207. This mapping is specified at the lowest level of standards' clauses – the level of tasks, providing the highest detail on the relationship. Since IEC 62304 was developed based on ISO/IEC 12207, the majority clauses of IEC 62304, seventy out of eighty-seven are mapped to the corresponding clauses of ISO/IEC 12207. Such a high-level of mapping enables to extend the existing mapping of ISO/IEC/IEEE 29119-2 with ISO/IEC 12207 to define the relationship with IEC 62304. The following section describes the approach to the mapping and identification of the relationships between ISO/IEC/IEEE 29119-2 and IEC 62304.

4.5.3 Mapping ISO/IEC/IEEE 29119-2 to IEC 62304

To define the related clauses of ISO/IEC/IEEE 29119-2 and IEC 62304, a mapping of Table B.2 from ISO/IEC/IEEE 29119-2 and Table C.5 from IEC 62304 was conducted. Figure 10 presents the first step of this mapping, which identifies ISO/IEC 12207 clauses that appear in both tables and are referred to as Mapped Clauses.

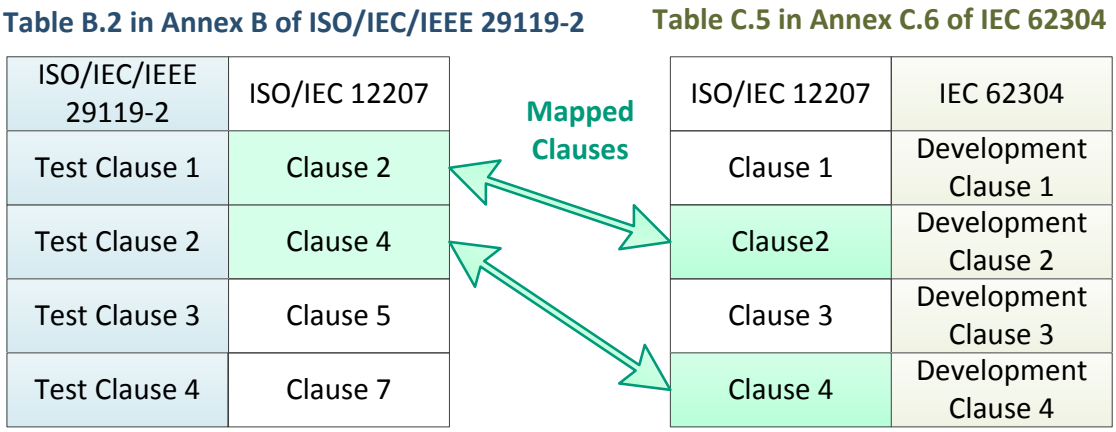


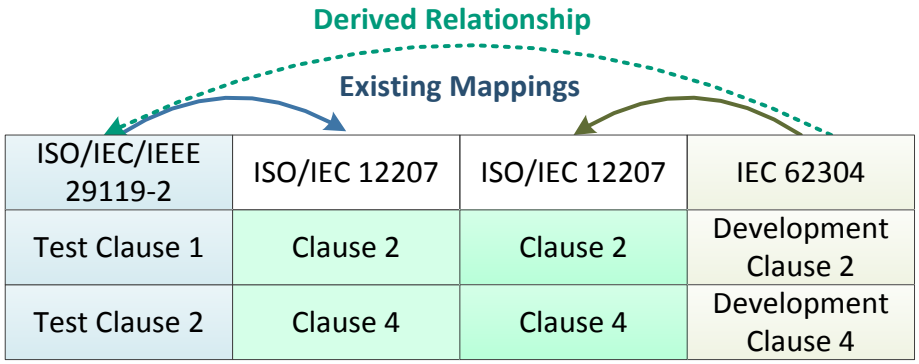
Figure 10 Identification of ISO/IEC 12207 Mapped Clauses

Since table B.2 maps test clauses of ISO/IEC/IEEE 29119-2 to corresponding clauses of ISO/IEC 12207, and these in turn in table C.5 are mapped to corresponding clauses of IEC 62304, these tables enable mapping ISO/IEC/IEEE 29119-2 clauses to corresponding clauses of IEC 62304. The ISO/IEC 12207 clauses mapped with both, IEC 62304 and ISO/IEC/IEEE 29119-2 clauses, are considered as *Mapped Clauses* between IEC 62304 and ISO/IEC/IEEE 29119-2 for the purpose of this study. ISO/IEC 12207 clauses related either to test or development clause indicate lack of relationship between ISO/IEC/IEEE

29119-2 and IEC 62304 clauses. For example, Clause 1 of ISO/IEC 12207 is not a *Mapped Clause*, because it appears only in table C.5, but not in table B.2.

In the second step of the mapping, the ISO/IEC 12207 *Mapped Clauses* were used to derive related IEC 62304 and ISO/IEC/IEEE 29119-2 clauses as presented in Table 4. The derived relationships for the ISO/IEC/IEEE 29119-2 and IEC 62304 clauses are those mapped to the same ISO/IEC 12207 *Mapped Clause*. For example, Table 4 presents that Test Clause 1 of ISO/IEC/IEEE 29119-2 is mapped to the Clause 2 of ISO/IEC 12207 and Development Clause 2 of IEC 62304 is mapped to the same Clause 2 of ISO/IEC 12207. These existing mappings allow a relationship to be derived between Test Clause 1 of ISO/IEC/IEEE 29119-2 and Development Clause 2 of IEC 62304.

Table 4 Sample Mappings of ISO/IEC/IEEE 29119-2 to IEC 62304



As an output of the mapping conducted, the mapping table was developed in an Excel file, a sample of which is presented in Table 5. This table with identified ISO/IEC 12207 *Mapped Clauses* and *Derived Relationships* between ISO/IEC/IEEE 29119-2 and IEC 62304 clauses is attached in Appendix A of the thesis.

Table 5 Sample of Mapping ISO/IEC/IEEE 29119-2 to IEC 62304

ISO/IEC/IEEE 29119-2 Test Clause	ISO/IEC 12207 Clause Process/ Activity/ Task	ISO/IEC 12207 Clause Process/ Activity/ Task	IEC 62304 Development Clause
7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.2 Monitor (TMC2)	***7.1.6 Software Integration 7.1.6.3.1 Software Integration 7.1.6.3.1.5***	***7.1.6 Software Integration 7.1.6.3.1 Software integration 7.1.6.3.1.5***	***5.1.6 Software Verification Planning***
7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.3 Control (TMC3)	6.3.2 Project Assessment and Control 6.3.2.3.2 Project Control 6.3.2.3.2.1	6.3.2 Project Assessment and Control 6.3.2.3.2 Project Control 6.3.2.3.2.1	5.1.2 Keep Software Development Plan Updated
7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.4 Report (TMC4)	6.1.2 Supply 6.1.2.3.4 Contract execution 6.1.2.3.4.15	N/A	N/A

For example, in the first row and first column of Table 5, the *7.3.4.2 Monitor* activity of *Test Monitoring and Control* process of ISO/IEC/IEEE 29119-2 is mapped in the second column to task *7.1.6.3.1.5* of the activity *7.1.6.3.1 Software Integration* as a sub clause of the process *7.1.6 Software Integration* of ISO/IEC 12207. This ISO/IEC 12207 task is mapped to the same task in the third column, and this in turn is mapped to IEC 62304 development clause *5.1.6 Software verification planning*. Since ISO/IEC 12207 task is mapped to both test activity *7.3.4.2 Monitor* and development task *5.1.6 Software verification planning*, it is a *Mapped Clause*. The high detail of the ISO/IEC 12207 *Mapped Clause* at the level of task enables to derive the relationship between test and development clauses in the first and fourth column. The same high detail of ISO/IEC 12207 *Mapped Clause* applies to the second row. For the development of the framework, this means that these ISO/IEC/IEEE 29119-2 test activities are directly related to corresponding IEC 62304 development clauses and these relationships with logical dependencies need to be defined in the MED-V-STEP framework. These test activities interact with related development activities, so their implementation needs to adhere to their logical dependencies.

Table 6 Sample of Mapping ISO/IEC/IEEE 29119-2 to IEC 62304

ISO/IEC/IEEE 29119-2 Test Clause	ISO/IEC 12207 Clause Process/ Activity/ Task	ISO/IEC 12207 Clause Process/ Activity/ Task	IEC 62304 Development Clause
7 Test Management Processes ***7.3 Test Monitoring and Control Process*** ***7.3.4.4 Report (TMC4)***	***6.1.2 Supply*** ***6.1.2.3.4 Contract execution*** ***6.1.2.3.4.15***	***N/A***	***N/A***

In another example in Table 6, for the *6.1.2.3.4.15* task of ISO/IEC 12207 there is no mapped clause in the third column. The lack of clauses in the third and fourth columns determines that *7.3.4.4 Report* activity of ISO/IEC/IEEE 29119-2 is not related to any development clause of IEC 62304. For the development of the MED-V-STEP framework, it means that the 7.3.4.4 *Report* activity of ISO/IEC/IEEE 29119-2 is not mapped in the framework to any development clause of IEC 62304. Testers can perform this activity without providing output or receiving input from a development activity. It means that if IEC 62304 activities were performed without ISO/IEC/IEEE 29119-2 activities, these testing best practice would not be known for implementation. To successfully maintain related activities, their logical dependencies must be determined, as described in Section 4.5.4.

4.5.4 Defining Logical Dependencies of Related Clauses

To fully define the relationship, the logical dependencies of related clauses need to be defined. Once the test clauses of ISO/IEC/IEEE 29119-2 and development clauses of IEC 62304 were mapped, they provided the basis for defining their logical dependencies. The logical dependence provides detailed information about the nature of the relationship between related clauses. The mapped standards' clauses were examined to define which clause generates the input or is a part of another clause. The examination of logical dependencies performed in this study involved a review of the content of the clauses and *Mapping Explanation* in Table B.1 in ISO/IEC/IEEE 29119-2. The review identified part of the content describing common issues addressed by both clauses and determined which activity is part of or input to another activity. An example in Figure 11, presents *"Organize Test Plan Development"* activity of ISO/IEC/IEEE 29119-2, which in accordance with this standard includes identification and notification of people involved in risk management for testing-related risks. This activity was therefore defined during the development of the MED-V-STEP framework as a part of a wider *Risk Management* activity of IEC 62304.

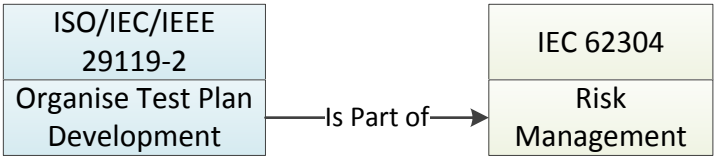


Figure 11 Logical Dependence of ISO/IEC/IEEE 29119-2 and IEC 62304 Activities

The inclusion of the output of the *Organise Test Plan Development* activity, which was defined as part of the related *Risk Management* activity, enhances the efficiency of their implementation. If the *Organise Test Plan Development* activity was applied in isolation, without following the logical dependency, there is a risk that the output of this activity would not be used as part of related *Risk Management* activity, and without affecting subsequent development activities. Logical dependencies in the MED-V-STEP framework fully define relationships between related clauses.

Defined in this study logical dependencies not only confirmed the previous mapping but also enhanced the ISO/IEC/IEEE 29119-2 software test clauses with the related IEC 62304 development clauses. The MED-V-STEP framework is intended for organisations who have implemented the requirements of IEC 62304, and determines which clause of IEC 62304 can be given more detail in terms of software testing best practice of ISO/IEC/IEEE 29119-2. This allows software testing to be conducted in line with current software testing best practice of ISO/IEC/IEEE 29119-2 while being related with the

requirements of IEC 62304 regarding software testing. Dependencies also define which activity generates output as input to and enhances related activity. Logical dependencies between related clauses that come from various standards were therefore defined and included within the MED-V-STEP framework. Section 4.6 describes the structure of the MED-V-STEP framework, which consists of ISO/IEC/IEEE 29119-2 test processes, whose test clauses are related to IEC 62304 development and ISO 14971 risk management clauses and their logical dependencies are defined. Examples of related clauses with defined logical dependencies are also given in Section 4.6.

4.6 MED-V-STEP Framework

The MED-V-STEP framework incorporates all test processes, activities and tasks of ISO/IEC/IEEE 29119-2 as well as IEC 62304 and ISO 14971 activities and tasks related to software testing. The framework was developed in an Excel file and the structure is consistent with ISO/IEC/IEEE 29119-2 process model as presented in Table 7.

Table 7 Sample of Table of Contents of MED-V-STEP Framework with Processes of ISO/IEC/IEEE 29119-2

N	MED-V-STEP Framework
OTP	Organisational Test Process - Introduction
OTP	Organisational Test Process - Overview, Purpose & Outcomes
OT1	Develop Organisational Test Specification
OT2	Monitor and Control Use of Organisational Test Specification
OT3	Update Organisational Test Specification
OT	Information items
TMP	Test Management Processes - Introduction
TPP	Test Planning Process - Overview, Purpose & Outcomes
TP1	Understand Context
TP2	Organize Test Plan Development
TP3	Identify and Analyse Risks
TP4	Identify Risk Mitigation Approaches
TP5	Design Test Strategy
TP6	Determine Staffing and Scheduling
TP7	Record Test Plan
TP8	Gain Consensus on Test Plan
TP9	Communicate Test Plan and Make Available
TP	Information items
TMCP	Test Monitoring and Control Process - Purpose & Outcomes
TMC1	Set-Up
TMC2	Monitor
TMC3	Control
TMC4	Report
TMC	Information Items
TCP	Test Completion Process - Purpose & Outcomes
TC1	Archive Test Assets

According to the findings of standards review in Section 2.4.2, the efficient application of software testing requires the implementation of ISO/IEC/IEEE 29119-2 generic software test process model in accordance with the logical dependencies of related clauses. To maintain high efficiency of software testing, the framework structure is consistent with the interrelated process groups, processes and activities of this standard, where the output of one activity generates the input of another related activity. The clauses presented in the table of contents are described in detail in separate Excel worksheets.

Figure 12 presents the test process worksheet, which provides a brief overview of the Organisational Test Process (OTP), its purpose, expected outcomes, and a diagram of process's activities as described in ISO/IEC/IEEE 29119-2.

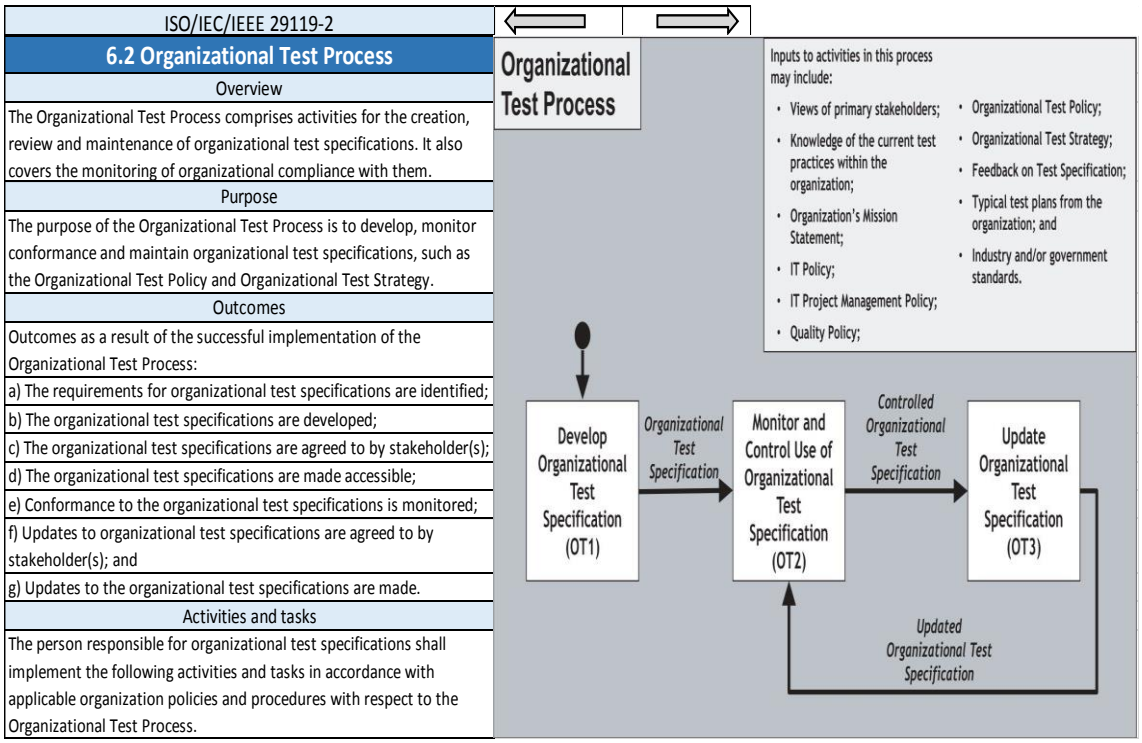


Figure 12 Worksheet with Test Process

From here, navigation arrows lead to worksheets with ISO/IEC/IEEE 29119-2 process activities. For each activity, all tasks of this activity are listed as described in ISO/IEC/IEEE 29119-2. When performing test activities that have not been mapped to any IEC 62304 development clause, such as, for example, some activities of *Test Planning Process*, testers neither provide test outputs as input to the development process nor use the outputs from the development process as input to testing. These test activities while not contained in IEC 62304 enhance IEC 62304 by providing information on software testing best practice.

For the ISO/IEC/IEEE 29119-2 test activity or task which was mapped to the related development clause of IEC 62304, the related clause is presented on the worksheet with this ISO/IEC/IEEE 29119-2 activity. The relationship of related clauses is explained by the description of the logical dependence developed by the researcher by a review of the content of the related clauses and *Mapping Explanation* in Table B.1 in ISO/IEC/IEEE 29119-2. The description refers to the common issue of these clauses and determines which activity is part of or generates input of a related activity.

The example of activity worksheet with ISO/IEC/IEEE 29119-2 test activities related to IEC 62304 development activity is given in Table 8. On the left side of the table are the test activities and tasks of OTP as described in ISO/IEC/IEEE 29119-2. On the right side of the table is the activity of IEC 62304 that has been identified by the researcher as related to test activities on the left side of the table as described in Section 4.5. The logical dependence of this relationship, developed by the researcher as part of the development of the framework, is described in the middle of the table between these related activities.

Table 8 Test Activity Related to Development Activity

	ISO/IEC/IEEE 29119-2	⇐ ⇒	IEC 62304
OT1	Develop Organizational Test Specification This activity consists of the following tasks:	Logical Dependence	5.1.7 Software Risk Management Planning
a)	Requirements for the organizational test specifications shall be identified from the current testing practices within the organization, from stakeholders and/or will be developed by other means. NOTE: This can be achieved by analyzing relevant source documents, through workshops, interviews or other suitable means.	OT1 + OT2 + OT3 shall include *"A definition of risk management processes for testing-related risks"* which is a part of 5.1.7	The manufacturer shall include or reference in the software development plan, a plan to conduct the activities and tasks of the software risk management process, including the management of risks relating to SOUP. [Class A, B, C] NOTE: See Clause 7 (Software risk management process)
b)	The organizational test specification requirements shall be used to create the organizational test specification.		
c)	Approval on the content of the organizational test specification shall be obtained from the stakeholders.		
d)	The availability of the Organization Test Specification shall be communicated to the stakeholders in the organization.		
OT2	Monitor and Control Use of Organizational Test Specification This activity consists of the following tasks:		
a)	Usage of the Organizational Test Specification shall be monitored to determine whether it is being used effectively within the		
b)	Appropriate actions shall be taken to encourage alignment of stakeholders to the organizational test specification.		
OT3	Update Organizational Test Specification This activity consists of the following tasks:		
a)	Feedback on use of the organizational test specification should be reviewed.		
b)	The effectiveness of the use and management of the organizational test specification should be considered and any feedback and changes to improve its effectiveness should be determined and approved. NOTE: This can be achieved by reviewing feedback, through workshops, interviews and other suitable means.		
c)	Where changes to the organizational test specification have been identified and approved, these changes shall be implemented.		
d)	All changes to the organizational test specification shall be communicated throughout the organization including to all stakeholders.		
OT	Information items As a result of carrying out this process, the following information item shall be produced:		
a)	Organizational Test Specification EXAMPLE: Organizational Test Policy, Organizational Test Strategy.		

In this example, *OT - Organisational Test Specification* activities of ISO/IEC/IEEE 29119-2 are related to the *5.1.7 - Software Risk Management Planning* activity of IEC 62304. This relationship was derived based on the common ISO/IEC 12207 *Mapped Clause* as described in Section 4.5.3. The description of the logical dependence of related clauses in the middle column is defined as part of the framework development as described in Section 4.5.4. The explanatory content in blue italics: *"A definition of the*

risk management process for testing-related risks" is based on following explanation in Table B.1 in ISO/IEC/IEEE 29119-2:

"The definition of risk management processes for testing is supported by clause 6.2 Organisational Test Process, which can be used to define organisational processes for managing and documenting testing-related risks." (2013b p. 43)

According to this explanation, *"A definition of the risk management process for testing-related risks"* is the output of OTP activities that were mapped in this study to the *5.1.7 - Software Risk Management Planning* activity of IEC 62304. As the testing-related risk is one of the possible software risks, this study determined that *"A definition of the risk management process for testing-related risks"* is part of a broader *"Software Risk Management Planning"*.

This relationship determines the need to provide the defined risk management process for testing-related risk to the software development team at the stage of software risk management planning. The inclusion of the output of these OTP activities affects risk management planning, as well as subsequent development activities. Such enhancement of test activities with the development activity enables their implementation in accordance with their relationships and thus increases the efficiency of their application.

Some of the related development clauses of IEC 62304 reference ISO 14971 Risk Management standard and require the corresponding clause of this standard to be implemented. In this case, the development of the framework in the description of the IEC 62304 clause also includes the description of the relevant ISO 14971 clause as shown in Table 9. In this example, *TP2 - Organize Test Plan Development* clause of ISO/IEC/IEEE 29119-2 is a part of *4.2 - Risk Management* clause of IEC 62304. This clause in the IEC 62304 standard refers to the risk management process according to ISO 14971, but does not provide detailed information on relevant related ISO 14971 activity or task. Also, Table C.2, annexed in IEC 62304, mapping ISO 14971 to IEC 62304, due to the low level of detail of the mapping, does not allow to derive the relationship between ISO/IEC/IEEE 29119-2 and ISO 14971. Therefore, it was necessary for researcher to examine the content of the ISO 14971 risk management clauses to identify the relationships with IEC 62304 and ISO/IEC/IEEE 29119-2 needed to develop the framework in this study.

Table 9 IEC 62304 Clause Referencing ISO 14971

	ISO/IEC/IEEE 29119-2	⇐ ⇒	IEC 62304
TP2	Organize Test Plan Development This activity consists of the following tasks:	Logical Dependence	4.2 Risk Management
a)	Based on the testing requirements identified in the Understand Context activity (TP1), those activities that need to be performed to complete test planning, shall be identified and scheduled.	TP2 shall include *"Identification and notification of people involved in risk management for testing-related risks"* which is a part of 4.2	The manufacturer shall apply a risk management process complying with ISO 14971: 3.2 Management responsibilities Top management shall provide evidence of its commitment to the risk management process by: – ensuring the provision of adequate resources and – ensuring the assignment of qualified personnel (see 3.3) for risk management.
b)	The stakeholders required to participate in these activities should be identified.		
c)	Approval of the activities, schedule and participants shall be obtained from the relevant stakeholders. Example 1: The Project Manager and/or Project Test Manager. NOTE: This could require repeating tasks a) and b).		
d)	Stakeholder involvement should be organized. Example 2: Request project manager to schedule a meeting for review of the test strategy		

In the given example in Table 9, the review of ISO 14971 identified corresponding *3.2 - Management Responsibilities* clause as related to *TP2* of ISO/IEC/IEEE 29119-2. The description of this clause was included in the description *4.2 - Risk Management* clause of IEC 62304 in the MED-V-STEP framework. As the *TP2* clause of ISO/IEC/IEEE 29119-2 was mapped to *4.2* clause of IEC 62304, which was subsequently mapped to *3.2* clause ISO 14971, *TP2* test clause interacts with the risk management process. Identification of people involved in risk management for testing-related risk under *TP2* of ISO/IEC/IEEE 29119-2 is part of assignment of qualified personnel for broader risk management in accordance with *3.2* of ISO 14971. Therefore, output of *TP2 - Organise Test Plan Development* was defined as part of *3.2 - Management Responsibilities*. The inclusion of the output of *Organisational Test Plan Development* activity extends the scope of *Management Responsibilities* activity, and thus affects subsequent risk management activities.

Another example in Table 10 presents how the ISO/IEC/IEEE 29119-2 test activities provide more detailed information on how to fulfil software testing related requirement according to IEC 62304 standard. In the left column of Table 10, *Test Design & Implementation* activities according to the ISO/IEC/IEEE 29119-2 standard, such as *TD2*, *TD3*, *TD4*, TD5 and *TD6*, cover the area of deriving test conditions, test coverage items, test cases, test sets and test procedures. Each of these activities consists of three to six tasks that describe how to perform the particular activity. All these *Test Design & Implementation* activities and tasks provide greater detail on how to fulfil the IEC 62304 software testing related requirement - *5.7.1 Establish Tests for Software Requirements*, which is presented in the right column of Table 10. In this way, software testing best practice of ISO/IEC/IEEE 29119-2 is aligned with and enhances the software testing requirement of IEC 62304.

Table 10 ISO/IEC/IEEE 29119-2 Clauses Providing Detailed Information on IEC 62304 Requirement

	ISO/IEC/IEEE 29119-2	⇐ ⇒	IEC 62304
TD2	Derive Test Conditions This activity consists of the following tasks:	Logical Dependence	5.7.1 Establish Tests for Software Requirements
a)	Based on the test completion criteria specified in the Test Plan, the test conditions for each feature shall be determined.	5.7.1 *"establish a set of tests for conducting software system testing"* is generated by outputs of TD2, TD3, TD4, TD5 and TD6 *"Derive Test: Conditions, Coverage Items and Cases. Assemble Test Sets and Derive Test Procedures."*	The manufacturer shall establish and perform a set of tests, expressed as input stimuli, expected outcomes, pass/fail criteria and procedures, for conducting software system testing, such that all software requirements are covered.
b)	The test conditions shall be prioritized using the risk exposure levels documented in the Identify and Analyze Risks activity (TP3).		
c)	The test conditions shall be recorded in the test design specification.		
d)	The traceability between the test basis, feature sets and test conditions shall be recorded.		
e)	The test design specification shall be approved by the stakeholders.		
TD3	Derive Test Coverage Items This activity consists of the following tasks:		
a)	The test coverage items to be exercised by the testing shall be derived by applying test design techniques to the test conditions to achieve the test completion coverage criteria specified in the Test Plan.		
b)	The test coverage items shall be prioritized using the risk exposure levels documented in the Identify and Analyze Risks activity (TP3).		
c)	The test coverage items shall be recorded in the test case specification.		
d)	The traceability between the test basis, feature sets, test conditions and test coverage items shall be recorded.		
TD4	Derive Test Cases This activity consists of the following tasks:		
a)	One or more test cases shall be derived by determining pre-conditions, selecting input values and, where necessary, actions to exercise the selected test coverage items, and by determining the corresponding expected results.		
b)	The test cases shall be prioritized using the risk exposure levels documented in the Identify and Analyze Risks activity (TP3).		
c)	The test cases shall be recorded in the test case specification.		
d)	The traceability between the test basis, feature sets, test conditions, test coverage items and test cases shall be recorded.		
e)	The test case specification shall be approved by the stakeholders.		
TD5	Assemble Test Sets This activity consists of the following tasks:		
a)	The test cases may be distributed into one or more test sets based on constraints on their execution.		
b)	The test sets shall be recorded in the test procedure specification.		
c)	The traceability between the test basis, feature sets, test conditions, test coverage items, test cases and test sets shall be recorded.		
TD6	Derive Test Procedures This activity consists of the following tasks:		
a)	Test procedures shall be derived by ordering test cases within a test set according to dependencies described by pre-conditions and post-conditions and other testing requirements.		
b)	Any test data and test environment requirements that are not already included in the Test Plan shall be identified.		
c)	The test procedures shall be prioritized using the risk exposure levels documented in the Identify and Analyze Risks activity (TP3).		
d)	The test procedures shall be recorded in the test procedure		
e)	The traceability between the test basis, feature sets, test conditions, test coverage items, test cases, test sets and test procedures (and/or		
f)	The test procedure specification shall be approved by the stakeholders.		

Using the MED-V-STEP framework, testers can apply a risk-based approach of ISO/IEC/IEEE 29119-2 to software testing as current software testing best practice adapted to the medical device software industry. The enhancement of the test processes of ISO/IEC/IEEE 29119-2 with related development and risk management clauses, determines IEC 62304 and ISO 14971 clauses that can be extended with software testing best practice according to ISO/IEC/IEEE 29119-2.

4.7 Summary and Conclusion

The research approach presented in this chapter enabled the design and development of the MED-V-STEP framework. Prior to the development of this framework, there was no consolidated resource on medical device software testing best practice.

An initial high-level mapping of ISO/IEC/IEE 29119-2 to IEC 62304 identified existing relationships between processes and activities of these standards and indicated the potential for their consolidation, which was confirmed by the Software Testing Organisation and international 11th Systems Testing and Validation Workshop. In response to this, it was decided to perform a detailed mapping to develop a software testing best practice framework for medical device software. The MED-V-STEP framework was developed and is attached in this thesis in Appendix B with the CD which contains the Excel file with the entire MED-V-STEP framework.

The MED-V-STEP framework consists of ISO/IEC/IEEE 29119-2 generic software test processes adapted to the IEC 62304 medical device software development and ISO 14971 risk management activities. Therefore, the framework may be beneficial in efficient implementation of software testing best practice for medical device software which is subject to framework validation. The ISO/IEC/IEEE 29119-2 generic software test processes provide current best practice, therefore, may be beneficial for medical device software development organisations in addressing the challenges of testing complex software systems and detecting increasing volume of defects.

The framework defines relationships between the ISO/IEC/IEEE 29119-2 test, IEC 62304 development and ISO 14971 risk management activities. By implementing the MED-V-STEP framework, organisations don't have to identify and implement activities from multiple standards and define their relationships. Therefore, the framework is proposed as potential way to realise benefits in efficient implementation of the required sets of software test activities at the appropriate development stage where the output of one activity is used as input to enhance the related activity.

The indicated benefits of the framework will only be proven through the framework validation. Chapter 5 presents the validation of the MED-V-STEP framework by industry professionals, in terms of its impact on the implementation of software testing best practice for medical device software and expected benefits of its implementation.

Chapter 5 Focus Group & Questionnaire Validation of MED-V-STEP Framework

In this thesis, an innovative MED-V-STEP framework was developed to address the insufficient evolution of medical device software testing. Since this study aimed to introduce an innovative software testing practice that solves the research problem in general rather than in a particular organisation, the validation focused on evaluating the concept of this innovative practice to see if it could be beneficial for industrial use. The purpose of the framework validation was to determine whether or not the framework contributes to the efficient implementation of software testing best practice for medical device software. In this study, focus groups and questionnaires were performed to validate the MED-V-STEP framework. The requirement of DkIT Academic Council and Institute Research Ethics Committee to obtain ethical approval for such validation is covered under the existing agreement of organisations involved in the validation process with the Irish Software Research Centre (DKIT 2015).

5.1 Introduction

For the purpose of this study, the appropriate DSR validation technique is the focus group for obtaining qualitative data and the questionnaire for obtaining quantitative data as outlined in the methodological Section 3.3.4. The focus groups carried out in this study cover both types of focus groups. The Exploratory Focus Group was used to elicit suggestions for improvement or refinement of the MED-V-STEP framework. The Confirmatory Focus Group was used to discuss the impact of the MED-V-STEP framework on improving software testing in the medical device domain in future implementation, as discussed in the next section. In order to carry out the focus group, the following preparatory steps were carried out, such as: formulation of the focus group's purpose, definition of the sample frame, identification of the moderator, development and initial testing of the questioning route and recruitment of participants. After the focus group was executed, the data obtained was analysed, interpreted and reported.

Questionnaire were also used to analyse the data obtained from focus groups. The questionnaire obtains quantitative data, which makes it possible to measure the data obtained. In this study, questionnaires were conducted using close-ended questions with scoring from 1 to 5 to measure participants' perception of the efficiency of the framework in improving medical device software testing practices. The following sections describe the approach to the performance of the validation.

5.2 Focus Group Preparation

5.2.1.1 Formulate Goal of Focus Group

The planning of the focus groups begins with the identification of the goals for the focus group. Referring to *RO.4 – Validate the framework in terms of providing information on the implementation of software testing best practice for the medical device software,* to answer *RSQ.4 Can the use of the framework contribute to the implementation of software testing best practice for the medical device software?*

the goal of the focus group has been defined as follows:

Obtain participant evaluation on the impact of the MED-V-STEP framework on the implementation of software testing best practice for the medical device software.

The evaluation addresses the efficiency of the MED-V-STEP framework in providing software testing best practice for medical device software, and whether this innovative practice could be beneficial for industrial use. The efficiency of the MED-V-STEP framework is defined as the ability to provide information on the joint implementation of generic software testing best practice in relation to medical device software processes, and to prevent these processes from being implemented twice or separately.

5.2.1.2 Identify the Sample Frame

For performing focus groups, two organisations have been identified, a Medical Device Software Development Organisation and a Software Testing Organisation.

A Medical Device Software Development Organisation develops software systems for Medical Device sector that are subject to regulatory compliance. The business partners of this organisation are some of the world's largest medical device and pharmaceutical companies. This organisation was involved because the focus of this study is on testing medical device software and related challenges. This organisation achieved certification to and follows processes and activities which fulfil the requirements of IEC 62304 and ISO 13485. Their experience in applying software development and risk management best practices according to these international standards and in achieving medical device regulatory compliance makes them suitable for validating software testing best practice for medical device software. The organization’s team consists of over 30 people, such as engineers, scientists and industrial designers. They can contribute to the framework's validation from the perspective of their experience of dealing with safety and compliance issues.

A Software Testing Organisation is an Irish branch with 350 employees of the world's leading multinational software quality group. This organisation has been involved in validation of the MED-V-STEP framework because this study deals with software testing and related challenges. The services provided by this organisation include software quality assurance and test management in a wide range of industries. This Organisation is currently implementing ISO/IEC/IEEE 29119-2, and is familiar with performing software testing in safety critical domains that are subject to regulatory requirements. The senior consultants and software testers of this organisation can contribute to validating the framework from the perspective of their experience in integrating domain-specific requirements within the generic test processes, as well as in testing embedded systems.

When considering one focus group for both organisations, the following advantages and disadvantages were considered. One focus group would bring together a higher number of participants with various qualifications and perspectives which would enable a broader range of discussion. However, due to commercial sensitivity, participants from one organisation may be biased due to the presence of participants from another organisation and reluctant to express their opinion openly. The sensitive information related to the software quality or testing challenges may result in concerns about expressing opinions and corporate reluctance to participate. This may bias the discussion and consequently, the evaluation of the MED-V-STEP framework. Medical Device Software Development Organisation in the presence of another organisation may not be willing to admit and discuss potential software quality issues and challenges related to software testing. This lack of willingness may be due to the fear that it will negatively affect the image of the organisation and its product. Therefore, two focus groups were conducted, *Focus Group 1* with the Medical Device Software Development organisation and *Focus Group 2* with the Software Testing Organisation.

Another reason for organising two focus groups concerns the different perspectives and priorities of the two organisations. Development organisations tend to favour development and to underestimate testing (Zelkowitz 2013). In this study, this tendency may apply to Medical Device Software Development Organisation that is involved in the framework validation. Their view of extending the development process with additional test processes using the MED-V-STEP framework may be influenced by the tendency to use the organisation's time and financial resources for development rather than software testing.

On the other hand, for Software Testing Organisation, any software quality issue and the need for testing are directly related to their professional activity. Software testing and software testing improvement are both the primary interest of the Software Testing Organisation. The intention to get an opinion from each organisation separately, not influenced by each other, supported the decisions of performing two focus groups.

A focus group usually involves four to twelve people who discuss the topic of interest (Tremblay et al. 2010). Seven participants were planned and attended the *Focus Group 1* with the Medical Device Software Development Organisation. This number represents the average number of anticipated participants. Therefore, it was considered sufficient for the purpose of performing the focus group. Two participants were planned and attended the *Focus Group 2* with Software Testing Organisation, which is below the advised range. However, this low number was determined by the constraints of the organisation. The inability to involve more participants is perceived as a limitation of this study. However, conducting the focus group with SQA Senior Consultants experienced in testing in safety critical domains was considered valuable due to their competence to contribute to the validation of the framework. The selection of applicants was based on their qualifications which are presented in Section 5.2.1.5.

5.2.1.3 Identify the Moderator

The role of the moderator is to promote interaction and keep the discussion on the topic of interest. The moderator needed to be familiar with the MED-V-STEP framework and be comfortable presenting it to the participants (Tremblay et al. 2010). As the framework is innovative and industry-specific, the researcher-designer is the person with the necessary knowledge and experience with a number of presentations about it. Given the necessary knowledge and practice in the presentations, in this study, the researcher-designer of the framework functioned as a moderator of the focus group.

However, the researcher acting as a moderator causes a possible bias of the focus group. The risk of bias due to researcher acting as a moderator was addressed by having an observer of the focus group (Tremblay et al. 2010). The role of the observer was not to actively participate in the focus group, but take the notes describing the course of the meeting and make note of any remarkable opinions or situations during the focus group. These notes recorded by the observer enable the moderator to stay on a line of questions, without having to worry about ensuring all of the information is noted correctly. The notes also provide assistance in preserving the data obtained, so that nothing important is left out.

A colleague from the research centre, having background in the area of testing and international standards was identified as an observer of the focus groups. This background was important because her own research concerned software testing, and in particular the taxonomy-based testing technique for medical device software. This involved the need to know the standards required for medical device regulatory compliance, as well as the issue of software testing in the medical device software industry and related challenges. Knowledge of software testing in the medical device domain based on her research, as well as general knowledge about this research, allowed her to understand the area discussed in focus groups, therefore she was identified as an observer of focus groups.

To ensure that no crucial information is missed during the course of the focus group, the discussion was recorded and the participants were advised that the recording was taking place. The digital recording of the focus group made it possible to retrospectively search the topics discussed and bring up the quoted statements on the matter.

5.2.1.4 Create and Pre-Test a Questioning Route

A set of questions was developed to obtain participant evaluation on the efficiency of the MED-V-STEP framework in the implementation of software testing best practice for medical device software. The questions for the focus group were ordered from the most general to the more specific. The first, general set consists of two questions about the efficiency of the framework in providing the necessary information to improve industrial practice. The goal of the first set of questions was to stimulate a discussion, starting with the role the framework should fulfil and its potential to improve industry practice. This part of the discussion formed the basis for raising more specific topics. Another, specific question is related to the relationships between software test and development processes which are important for their enhancement and efficient implementation. The aim of this question was to stimulate a discussion on the role of relationships defined within the framework for the enhancement of medical device software development life-cycle with generic software testing best practice. And finally, the last question is about suggested refinements of the framework.

A pilot presentation was carried out where these questions were discussed with colleagues from the research centre. A feedback and comments obtained helped to improve the set of questions in terms of their relevance, and the final version of the questions was defined. The final list of questions used to gain the evaluation of the MED-V-STEP framework is presented as follows:

A. Questions related to the overall structure of the framework:
    1) How efficient do you think the MED-V-STEP framework is in providing information on needed processes, activities and tasks and their relationships?
    2) How efficient do you think the MED-V-STEP framework is in improving industry software testing practices?
B. Question related to the relationships between test and development activities:
    3) How efficient do you think the MED-V-STEP framework is in defining how software test activities of ISO/IEC/IEEE 29119-2 need to be enhanced with the requirements of IEC 62304?
C. Question related to the suggested improvements of the MED-V-STEP framework:
    5) Can you tell me anything that you would do different or what would you change in the MED-V-STEP framework?

5.2.1.5 Recruit Participants

Two focus groups were performed, *Focus Group 1* with Medical Device Software Development Organisation and *Focus Group 2* with Software Testing Organisation.

The primary focus of the MED-V-STEP framework validation was on the review by Medical Device Software Development Organisation, as this is the domain in which the framework is intended to be implemented. The Medical Device Software Development Organisation follows processes and activities which fulfil the requirements of IEC 62304. Participants from Medical Device Software Development Organisation identified for performing focus group include one Senior Design QA Engineer, two QA Engineers, one Senior Software Developer, two Software Developers and one Electronics Design Engineer, seven participants in total. Knowledge of the requirements of the IEC 62304 standard enabled QA engineers and software developers from the company to assess the efficiency of enhancing software development process of IEC 62304 with test processes of ISO/IEC/IEEE 29119-2. Participants were also well placed to examine the relevance of the defined logical dependencies of related software test, development and risk management clauses.

As the MED-V-STEP framework is related to software testing and SQA, another important part of the framework validation was the review of the framework by the Software Testing Organisation. This organisation, from which two participants have been identified for validating the framework, has previously collaborated with a researcher by providing feedback on the progressing development of the MED-V-STEP framework.

Prior to the focus group, representative from this organisation attended researcher's presentations about progressing framework development and provided opinion on the framework and the development approach.

The Software Testing Organisation is currently implementing ISO/IEC/IEEE 29119-2, and is considering expanding its services to include software testing in the medical device domain. Due to their testing experience in various industries such as automotive domain which is safety critical similarly to the medical device domain, they are familiar with integrating domain-specific requirements within the generic test processes. From this organisation two QA Senior Consultants with software testing experience in similar safety critical domains, knowledge of ISO/IEC/IEE 29119-2 and this study were identified as participants of the focus group.

The larger number of participants is from the Medical Device Software Development Organisation and they are better placed to review the framework at this stage. However, the level of the experience in the field enabled participants from Software Testing Organisation comment on performing testing with the use of ISO/IEC/IEE 29119-2 and integrating domain-specific requirements. Also, the study and approach to the development were presented to a larger testing audience at the international 11th Systems Testing and Validation Workshop.

5.3 Validation by Medical Device Organisation

5.3.1 Conducting Focus Group 1

The *Focus Group 1* performed with participants of the Medical Device Software Development Organisation was carried out at the premises of the participating organisation. Two hours were planned for performing the focus group.

5.3.1.1 Presentation and Discussion on the Overall Structure of the MED-V-STEP Framework

To ensure the relevant discussion of the participants based on the good understanding of the framework, the overall structure of the MED-V-STEP framework was presented before the discussion.

The participant's discussion following the presentation was based on the list of questions presented in Section 5.2. The first two questions on the overall structure of the framework were discussed in turn. To encourage participants to take part in discussion, the moderator explained that the focus group's goal is to clarify the opinion through a joint discussion, ensuring that the discussion does not require the participant to have a precise opinion. In response, the participants began to take an active part in the discussion. When considering the first question from the list of questions on the overall structure of the framework, the participants raised questions about the role of international standards used in the framework. The questions raised gave an opportunity to clarify some of the participants' assumptions. The following are questions that were raised in relation to the use of ISO/IEC/IEEE 29119-2 within the MED-V-STEP framework.

The first question of the participants concerned: *"which ISO/IEC/IEEE 29119-2 clauses were included in the MED-V-STEP framework and which were not"*? Another similar question considered in more detail: *"does the MED-V-STEP framework includes only those clauses of ISO/IEC/IEEE 29119-2, which were mapped to IEC 62304 clauses, and others not"*? These questions were addressed by the moderator's explanation that the MED-V-STEP framework incorporates all clauses of the ISO/IEC/IEEE 29119-2 standard regardless of whether they are mapped to IEC 62304 development clauses or not. As it is a software testing framework for medical device software, the role of the framework is to inform medical device software testers not only of existing relationships between test and development activities, but also of all test clauses according to ISO/IEC/IEEE 29119-2. The framework structure is consistent with the interrelated processes, activities and tasks

of ISO/IEC/IEEE 29119-2 to provide current software testing best practice and to maintain its efficient implementation.

After this clarification, another question of participants arose: *"since the MED-V-STEP framework also includes those clauses of ISO/IEC/IEEE 29119-2 that were not mapped to IEC 62304, why were they included"*? The moderator explained that the inclusion of only those clauses that have been mapped to IEC 62304 would highlight existing relationships between ISO/IEC/IEEE 29119-2 and IEC 62304, but would not provide comprehensive information on software testing best practice as defined in ISO/IEC/IEEE 29119-2. The framework contains all clauses of ISO/IEC/IEEE 29119-2 to provide software testing best practice that is not specified in IEC 62304. The mapping of related test and development clauses is provided to extend software testing requirements of IEC 62304 with software testing best practice of ISO/IEC/IEEE 29119-2.

The next questions of participants were related to the use of IEC 62304 within the MED-V-STEP framework. One question considered: *"whether the MED-V-STEP framework covers all IEC 62304 clauses"*? The next question extended the previous one: *"if the MED-V-STEP framework does not cover all IEC 62304 clauses, why not"*? These questions are understandable for software engineers following the software life-cycle processes of IEC 62304 to develop software in compliance with medical device regulations. They would rather see mapping from the development stage to the related test activity.

To address these questions, the moderator explained that ISO/IEC/IEEE 29119-2 was identified as current software testing best practice standard that addresses the software testing gap coming from IEC 62304. In accordance with ISO/IEC/IEEE 29119-2, maintaining high efficiency of software testing requires the implementation of ISO/IEC/IEEE 29119-2 test clauses in accordance with their logical dependencies. To maintain the structure of interrelated clauses of ISO/IEC/IEEE 29119-2, this standard was taken as a starting point for the mapping and development of the MED-V-STEP framework. In addition, the use of the MED-V-STEP framework is intended for testers from medical device software development organisations that already fulfil the requirements of the IEC 62304 standard. For these organisations there is no need to duplicate IEC 62304 clauses that have already been implemented. The IEC 62304 development clauses, which have been included in the framework, are necessary to provide software testers with information on the test activities, where the direct relationship with development activities takes place.

Another raised question was: *"why the structure of the MED-V-STEP framework is not based on IEC 62304 but on ISO/IEC/IEEE 29119-2"*? This question was addressed by moderator with explanation that the aim of the framework is to enable efficient implementation of test processes of ISO/IEC/IEEE 29119-2 adapted for testing medical device software. ISO/IEC/IEEE 29119-2 defines the logical dependencies between related clauses and determines the sequence in which the related clauses need to be performed as a basis for their efficient implementation. Therefore, the framework structure is based on ISO/IEC/IEEE 29119-2, which is an international standard that has been developed based on state of the art and consensus among software testing practitioners who are nominated national experts and as such recognised as experts in this field.

The explanations provided by the moderator helped the participants to better understand how the MED-V-STEP framework was developed. This understanding was essential and helped them express their views not only on the efficiency of the MED-V-STEP framework in providing information on needed processes and their relationships, but also on improving software testing practice in the industry. Participants' assessments resulting from the discussion of the first two questions are presented in section 5.3.2.

5.3.1.2 Presentation and Discussion on Relationships between Test and Development Activities

After discussing the overall design of the framework, mapping and logical dependencies of related clauses were presented. The relationship of *Organisational Test Specification* activities with *Software Risk Management Planning* activity presented in Table 11 was provided as an example to the participants of the focus group. This example of the related clauses represents the structure of other relationships since, for all relationships defined in the MED-V-STEP framework, there are related clauses and their logical dependence. A detailed explanation of the mapping process and definition the logical dependence of mapped clauses in the given example was provided. The left column of the table contains the test activities of ISO/IEC/IEEE 29119-2: *OT1 Develop Organisational Test Specification, OT2 Monitor and Control Use of Organisational Test Specification, and OT3 Update Organisational Test Specification*. These test activities were mapped and defined as a part of the development activity of IEC 62304: *5.1.7 Software Risk Management Planning,* presented in the right column of the table.

Table 11 Relationship of Software Test and Development Clauses

	ISO/IEC/IEEE 29119-2	⇦ ⇨	IEC 62304
OT1	Develop Organizational Test Specification This activity consists of the following tasks:	Logical Dependency	5.1.7 Software Risk Management Planning
a)	Requirements for the organizational test specifications shall be identified from the current testing practices within the organization, from stakeholders and/or will be developed by other means. NOTE: This can be achieved by analyzing relevant source documents, through workshops, interviews or other suitable means.	OT1 + OT2 + OT3 shall include *"A definition of risk management processes for testing-related risks"* which is a part of 5.1.7	The manufacturer shall include or reference in the software development plan, a plan to conduct the activities and tasks of the software risk management process, including the management of risks relating to SOUP. [Class A, B, C] NOTE: See Clause 7 (Software risk management process)
b)	The organizational test specification requirements shall be used to create the organizational test specification.		
c)	Approval on the content of the organizational test specification shall be obtained from the stakeholders.		
d)	The availability of the Organization Test Specification shall be communicated to the stakeholders in the organization.		
OT2	Monitor and Control Use of Organizational Test Specification This activity consists of the following tasks:		
a)	Usage of the Organizational Test Specification shall be monitored to determine whether it is being used effectively within the		
b)	Appropriate actions shall be taken to encourage alignment of stakeholders to the organizational test specification.		
OT3	Update Organizational Test Specification This activity consists of the following tasks:		
a)	Feedback on use of the organizational test specification should be reviewed.		

This part of the discussion was based on another question related to the relationships between test and development activities. The aim was to gain participant's feedback about the efficiency of the defined relationships in determining which development activities could be extended with required sets of software test activities, and how the output of one activity could be used to enhance the related activity. Before expressing their opinions, the participants asked for further clarification on how the logical dependence of the related clauses was defined and how could one clause be enhanced by another clause?

The moderator readdressed the issues raised with the following explanation based on Table 11. This study identified that the process for managing testing-related risk is a part of a broader process for managing software risk. The logical dependence was defined as follows: *OT1, OT2 and OT3 shall include a definition of the risk management process for testing-related risks which is a part of 5.1.7 Software Risk Management Planning*. In this way, the *Organisational Test Specification* clauses of ISO/IEC/IEEE 29119-2 enhance the *Software Risk Management Planning* clause of IEC 62304. If these activities were applied separately, there is a risk that the *Organisational Test Specification* activities would not become part of the *Software Risk Management Planning* activity and without impacting on subsequent activities.

This example applies to all other relationships defined in the MED-V-STEP framework. For all relationships, if related clauses are applied in isolation, without following their logical dependence, they will not achieve enhancement. Organisations that develop medical device software in compliance with IEC 62304 conduct activities, such as

requirements analysis or design that generate the data as input needed for ISO/IEC/IEEE 29119-2 test planning process. Defined logical dependencies in the MED-V-STEP framework describe how to use this existing IEC 62304 output to enhance ISO/IEC/IEEE 29119-2 test process. Without this enhancement, there is a risk for related activities that the output of one activity would not be provided and without affecting the related activity. The explanation provided by the moderator helped to clarify assumptions of the participants and to express their opinion in response to the questions discussed. The output of participants' assessment of the MED-V-STEP framework is presented in the Section 5.3.2 describing the focus group outputs.

5.3.1.3 Discussion on improving the MED-V-STEP framework

At the end of the focus group, participants discussed what would they do different or what would they change in the MED-V-STEP framework. During this part of the discussion based on the last question, they proposed improvements and refinements to the MED-V-STEP framework. As presented in Section 5.3.2, the proposed improvements related to tailoring the framework to the needs of the organisation for future implementation.

5.3.2 Findings from Focus Group 1

This section presents the findings of the *Focus Group 1* performed with Medical Device Software Development Organisation. The day after the focus group, the moderator and observer met to summarize their findings, providing collected data from the focus group. The data was collected by recording a focus group and noting feedback from participants. The observer also shared her impressions of direct observation of the focus group participants. At this meeting, no differences in the data collected were identified and consensus was reached that the data was collected accurately. The presentation of the results is organized in accordance with the list of predetermined questions.

5.3.2.1 Assessment of Overall Structure of MED-V-STEP Framework

When discussing the first question: *"How efficient do you think the MED-V-STEP framework is in providing information on needed processes, activities and tasks and their relationship?"*, one participant from the first focus group with a Medical Device Software Development Organisation reported:

> *"As a concept, providing both streams of standards is good and providing the mapping between ISO/IEC/IEEE 29119-2 to IEC 62304 is very useful and helpful for adoption."*

In this statement he confirmed that the standards selected for the development of the framework and the mapping of these standards performed are relevant. He also reported on second question: *"How efficient do you think the MED-V-STEP framework is in improving industry software testing practice?"*, stating that:

> *"From the quality point of view, I can see the potential for the improvement, because you use IEC 62304 for compliance of additional processes that are testing oriented, and they are obviously good practices and good standards to follow, so from the quality point of view, there is an improvement"*.

This statement confirms that the ISO/IEC/IEEE 29119-2 standard provides current testing best practice and therefore the potential of the framework in improving the quality of software testing. Another participant from the first focus group also confirmed the efficiency of the MED-V-STEP framework in providing needed processes and defining their relationships affirming that:

> *"The MED-V-STEP framework is very useful to fill testing gaps coming from IEC 62304."*

In the statement above, the participant with knowledge of the IEC 62304 standard and software testing in the medical device domain confirmed the relevance of this study in two respects. He confirmed the validity of the standards review in Section 2.4.2 as regards to the identified gap in IEC 62304 for detailed information on testing. He also confirmed the validity of developing the MED-V-STEP framework to eliminate this testing gap, as presented in Chapter 4. The next statement of another participant confirmed the efficiency of the MED-V-STEP framework in providing relationships between test and development processes.

> *"Benefit is that you have essentially mapped ISO/IEC/IEEE 29119-2 to IEC 62304 with as little overhead as possible.*

This statement confirmed the validity of the selection of the above standards and the mapping conducted to identify the relationships between them. In the following statement, the participant recognised the potential of the MED-V-STEP framework to:

> *"Improve the quality of software testing by performing to additional standards around software testing on top of IEC 62304."*

The selection of ISO/IEC/IEEE 29119-2 was considered relevant according to the statement of another participant:

"It looks reasonable useful adopting ISO/IEC/IEEE 29119-2 for software testing."

The participants also confirmed the potential of the MED-V-STEP framework to improve software testing efficiency in the medical device domain:

"The answer to the question of whether this idea is good for improving software testing practices is yes, absolutely, because you add additional test processes as a guarantee of test quality."

This statement confirms the efficiency of the framework to improve medical device software testing practice and the resulting benefit for organisations in the form of improved testing quality.*"*

5.3.2.2 Assessment of Relationships between Test & Development Activities

When discussing with Medical Device Software Development Organisation the question: *"How efficient do you think the MED-V-STEP framework is in defining how software test activities of ISO/IEC/IEEE 29119-2 need to be enhanced with the requirements of IEC 62304?"*, one participant confirmed the validity of the approach to mapping standards in order to identify related test and development clauses, stating:

"Identification of relevant testing and additional testing clauses at development stages is always good."

The following statement relates to identified relationships and defined logical dependencies determining what needs to be done:

"MED-V-STEP framework exposes relationships and shows exactly what of ISO/IEC/IEEE 29119-2 needs to be done."

This statement confirmed the validity of defined relationships and logical dependencies as useful for the efficient implementation of software testing for medical device software. Another participant confirmed that:

"The MED-V-STEP framework provides compliance of ISO/IEC/IEEE 29119-2 with IEC 62304."

This statement validates that the use of the MED-V-STEP framework in no way compromises the compliance with IEC 62304, and that the ISO/IEC/IEEE 29119-2 test processes are based on IEC 62304 requirements regarding software testing.

5.3.2.3 Suggested Improvements of MED-V-STEP Framework

A discussion to gain feedback on suggested improvements was conducted at the end of the focus group session. When discussing the question: *"Can you tell me anything that you would do different or what would you change in the MED-V-STEP framework?"*, some but not all of the software engineers would prefer that the structure of the MED-V-STEP framework not be based on ISO/IEC/IEEE 29119-2 but rather be based on IEC 62304 enhanced with processes of ISO/IEC/IEEE 29119-2. This perception of the MED-V-STEP framework is understandable for software engineers who are familiar and using the software life-cycle processes from IEC 62304 to develop their software. A framework based on this structure and including whole IEC 62304 would enable them to see mapping from the development perspective and make it easier for them to identify the relevant development stage with direct relationship with a testing activity.

From the research perspective in this study, the decision to perform mapping based on ISO/IEC/IEEE 29119-2 structure was determined by the research approach to identify the generic non-domain-specific software testing best practice and apply it in a safety critical domain - the medical device domain. This application has been achieved by adapting the generic software testing model of ISO/IEC/IEEE 29119-2 to standards for medical device software. This adaptation was done by mapping and consolidating ISO/IEC/IEEE 29119-2 clauses with related IEC 62304 and ISO 14971 clauses into a framework. The framework follows the sequence of software testing best practice determined by the structure of ISO/IEC/IEEE 29119-2, and relates these practices to IEC 62304, to ensure the efficient implementation of software testing best practice in the development of medical device software. Such a consolidated framework for medical device software testing has not been developed prior to this study. From the implementation perspective, a framework structure based on IEC 62304 would need to include all clauses of this standard and its implementation would duplicate the already implemented IEC 62304 processes. QA engineers from Medical Device Software Development Organisation understood that the MED-V-STEP framework is intended as a route to implementation of software testing best practice for organisations already compliant with IEC 62304. As a result of this understanding, they expressed their agreement with the approach of a framework not duplicating the requirements of the IEC 62304 standard but following the structure of the ISO/IEC/IEEE 29119-2 standard.

The participants from the Medical Device Software Development Organisation suggested that for the actual use of the MED-V-STEP framework, further expert review would raise

the validity of the framework. The validation in this study was conducted in a targeted way to validate the MED-V-STEP framework to a level where this concept could be proved. Further validation would be carried out on the same basis but in more depth, and how this would be done is outlined in the future work Section 6.6. Further expert review and pilot implementation of the MED-V-STEP framework would be required prior to the actual implementation and the use of the framework in the medical device software industry. However, this study developed an innovative framework, demonstrated its potential benefits for future industrial use, and defined a process for its further development. This software testing best practice framework for medical device software is based on established internationally recognised standards that have been consolidated to enable their efficient combined implementation.

Another suggestion concerned the adoption of the MED-V-STEP framework in practice. One participant identified the need for the development of supporting resources to aid the adoption of the MED-V-STEP framework in its next phase of development. In his opinion, the development of an action plan for the implementation of the framework would be beneficial. In response to this suggestion, the usability of the framework would form the focus of further development work.

Currently, the MED-V-STEP framework is in the form of a spreadsheet that is searched manually. Participants suggested that future development should be in the form of a searchable database. Again, this suggestion would form the focus of further development work and could include a user interface tailored to the needs of a particular company.

All suggestions raised are implementation-based, so they are outside the scope of this study as the aim of this study was the development of a framework for proof of concept. These suggestions, however, indicate opportunities for the further development and tailoring of the framework to the needs of a particular organisation in the future implementation.

5.3.3 Findings from Questionnaire 1

After completing the *Focus Group 1*, *Questionnaire 1* was carried out with their participants to obtain quantitative data on the previously discussed questions. While there were seven participants in the focus group, one participated by phone and therefore could not complete the questionnaire. So only six participants completed the questionnaire. Quantitative data contributes to understanding the extent to which participants perceive that the framework can contribute to the efficient implementation of software testing best practice in the medical device domain. For the questionnaires the questions formulated for the focus groups were used. Participants evaluated the answers in the scoring range from 1 (the lowest level) to 5 (the highest level) of the efficiency of the framework in improving medical device software testing practice.

Quantitative data have been obtained from questionnaire to give a measure to the questions discussed in the focus groups, and are presented in Appendix C. This data demonstrates a high-level of the efficiency of the MED-V-STEP framework for software testing in the medical device domain. The average assessment of participants is 3.8 in the scoring range of 1 to 5, in the areas covered by the questions:

How efficient do you think the MED-V-STEP framework is in providing information on needed processes, activities and tasks and their relationship?

Evaluation = 3.8 out of 5

How efficient do you think the MED-V-STEP framework is in improving industry software testing practice?

Evaluation = 3.8 out of 5

How efficient do you think the MED-V-STEP framework is in defining how software test activities of ISO/IEC/IEEE 29119-2 need to be enhanced with the requirements of IEC 62304?

Evaluation = 3.7 out of 5

These data demonstrating the high-level efficiency of the framework, which was positively assessed earlier in *Focus Group 1*, further confirm the validity of the framework. The summary of focus groups and questionnaires findings and conclusions are provided in Section 5.5. The next section presents validation of the MED-V-STEP framework by Software Testing Organisation which extends the findings from Medical Device Development Organisation presented in this section.

5.4 Validation by the Software Testing Organisation

5.4.1 Conducting Focus Group 2

The *Focus Group 2* with the Software Testing Organisation was carried out at the premises of the organisation. The structure and timeline of the *Focus Group 2* was the same as the first one, consisting of two hours planned for performing presentations followed by discussions.

5.4.1.1 Presentation and Discussion on the Overall Structure of the MED-V-STEP Framework

To ensure the relevant discussion of the participants, the overall structure of the MED-V-STEP framework was presented before the discussion.

The participant's discussion following the presentation was based on a list of questions presented in Section 5.2. First, the overall structure of the MED-V-STEP framework was discussed. When discussing the relevance and limitations of standards used for the development of the MED-V-STEP framework, one participant stated regarding the limitations of IEC 62304 in providing detailed description of software testing activities:

> *"That is interesting when you'll think that there would be a much tighter guideline around exactly what you have to do, to be compliant. But it is just, you need to do, but it is up to you."*

These opinions confirmed the need for the approach taken in this study to extend IEC 62304 requirements related to software testing with more detailed description of software testing. When discussing the applicability of ISO/IEC/IEEE 29119-2 to the software development life-cycle model of IEC 62304, one participants pointed:

> *"You are trying to marry the two together, the generic standard for software testing with the very specific medical device software standard."*

This comment indicated participants' good understanding of the approach to the development of the framework. When discussing the relevance of ISO/IEC/IEEE 29119-2 to address the testing gap in IEC 62304, another opinion expressed by the participants regarding applying ISO/IEC/IEEE 29119-2 processes from the early software development life-cycle stages of IEC 62304 was that:

> *"One of the challenges historically with software testing is that it came very late in software development life-cycle (SDLC). From my last 10 years' experience, testing is shifting left in SDLC, trying to embed testing*

as early as possible into SDLC and identify defects closer to the design and development stage. The later you do it, the more expensive it is to move back and correct and redo it."

This perspective has validated the benefits of using the ISO/IEC/IEEE 29119-2 processes within MED-V-STEP, as this standard advocate implementing test activities from the early life-cycle stages.

5.4.1.2 Presentation and Discussion on Relationships between Test and Development Activities

After discussing the overall structure of the MED-V-STEP framework, the example in Figure 13 used in the previous focus group of defined relationship between ISO/IEC/IEEE 29119-2 test activity and IEC 62304 development activity was provided. This given example is representing the structure of other related clauses and their logical dependence. As with the first focus group, a detailed explanation of the mapping process and definition the logical dependence of mapped clauses was provided. The *OT1*, *OT2*, and *OT3* test activities of ISO/IEC/IEEE 29119-2 were mapped and defined as a part of the *5.1.7 Software Risk Management Planning* activity of IEC 62304.

Regarding the example of IEC 62304 *Software Risk Management Planning*, participants raised the following question:

"That looks like very general statement in the IEC 62304 standard, almost an umbrella term for testing software, this whole software risk management. So, does it map just to those particular ISO/IEC/IEEE 29119-2 items, or does it map in general to everything in software development?"

To answer this question, the moderator explained that testing risk management is part of larger risk management process throughout the whole IEC 62304 SDLC. The participants' assessment of the MED-V-STEP framework is presented in the following section.

5.4.2 Findings from Focus Group 2

The day after the *Focus Group 2*, the moderator and observer summarized the findings, providing data collected by recording, remembering feedback from participants, impressions of direct observation of participants and taking notes. The presentation of the results in the following sub-sections is organized in accordance with the list of predetermined set of questions.

5.4.2.1 Assessment of Overall Structure of MED-V-STEP Framework

When discussing the first question from the list of predetermined questions: *"How efficient do you think the MED-V-STEP framework is in providing information on needed processes, activities and tasks and their relationships?"*, they assessed the efficiency in providing needed processes and defining their relationships favourably:

> *"Definitely, getting down to the lower level of detail in terms of the testing processes and such granularity is incredibly useful and really helps people to understand what they need to do."*

This statement of the QA Senior Consultant with knowledge of ISO/IEC/IEEE 29119-2 and experience with software testing in automotive domain confirms the relevance of the use of ISO/IEC/IEEE 29119-2 processes within the MED-V-STEP framework and the efficiency of the framework in providing the description of sets of test activities that need to be performed. The efficiency of the MED-V-STEP framework in defining the relationships between ISO/IEC/IEEE 29119-2 test activities and IEC 62304 development activities was assessed by participants as follows:

> *"The framework relates testing directly with IEC 62304 requirements"*

This feedback confirms the validity of these relationships developed in this study.

When discussing the question: *"How efficient do you think the MED-V-STEP framework is in improving industry software testing practice?"*, one participant stated that:

> *"Looking at what you are saying about the IEC 62304 standard, which says that you need to do software testing, and does not tell you how to do it, so providing this level of detail can be very useful."*

This opinion is similar to the opinion given in the other focus group and confirms the impact of the MED-V-STEP framework on improving software testing practice in the future industrial use.

5.4.2.2 Assessment of Relationships between Test & Development Activities

When discussing with Software Testing Organisation the question: *"How efficient do you think the MED-V-STEP framework is in defining how software test activities of ISO/IEC/IEEE 29119-2 need to be enhanced with the requirements of IEC 62304?"*, one participant stated that:

"Ensuring, that these two processes are related, is useful. The defined relationships allow testing and development to work together and the synchronisation between the two."

This feedback confirms the efficiency of the MED-V-STEP framework in enhancing ISO/IEC/IEEE 29119-2 test processes with IEC 62304 requirements to enable the effective collaboration between test and development activities. The framework was generally positively assessed in terms of that:

"The framework definitely looks like a worthwhile in effort, particularly when it is something high-level in terms of your area of expertise that will enable to map it into something existing. It is fantastic, it stops you to having to do everything from scratch."

5.4.2.3 Suggested Improvements of MED-V-STEP Framework

When discussing any improvements of the MED-V-STEP framework, the suggestion raised by two participants from the Software Testing Organisation was that the framework should include endpoints for signing off the completed software test activities to confirm that they were performed for a particular development stage. This is addressed by the requirements of ISO/IEC/IEEE 29119-2 which mandate reporting the completion of the test activities. Also, this suggestion is related to the practical application of the framework and the sign-off endpoints could be addressed by a checklist of tasks implemented as a spreadsheet or database. No more improvements were suggested to the framework, as the participants expressed as follows:

"The approach looks really good; it looks like it is something that can be followed. There is a lot of work done on the Excel version of the framework. I cannot see anything right now that requires change."

This feedback confirms the relevance of the stage to which the framework has been developed to date and its potential for future industrial use.

5.4.3 Findings from Questionnaire 2

After completing the *Focus Group 2*, *Questionnaire 2* was carried out with their participants to obtain quantitative data on the previously discussed questions and to contribute to understanding the extent to which participants perceive the framework efficiency in the implementation of software testing best practice in the medical device domain. For the questionnaires were used the questions formulated for the focus groups. Participants evaluated the answers in the scoring range from 1 (the lowest level) to 5 (the highest level) of the efficiency of the framework in improving medical device software testing practice.

Quantitative data have been obtained from questionnaires to give a measure to the discussed questions on focus groups, and are presented in Appendix C. This data demonstrates a highest-level of the efficiency of the MED-V-STEP framework for software testing in the medical device domain. The average assessment of participants is 5 in the scoring range of 1 to 5, in the areas covered by the questions:

How efficient do you think the MED-V-STEP framework is in providing information on needed processes, activities and tasks and their relationship?

Evaluation = 5.0 out of 5

How efficient do you think the MED-V-STEP framework is in improving industry software testing practices?

Evaluation = 5.0 out of 5

How efficient do you think the MED-V-STEP framework is in defining how software test activities of ISO/IEC/IEEE 29119-2 need to be enhanced with the requirements of IEC 62304?

Evaluation = 5.0 out of 5

These data demonstrating the highest-level of efficiency of the MED-V-STEP framework further confirm the validity of the framework which was positively assessed earlier in focus group. However, the high rating is limited in the following aspects. The low number of two participants could have influenced the limited discussion of the focus group which preceded the questionnaire. With a larger number of participants, more opinions could be presented, which could influence the final rating. Participants had experience in testing safety-critical software in other domains, but the evaluation may have been also affected by limited knowledge of requirements for medical device software. The summary of focus groups and questionnaires findings and conclusions are provided in the following section.

5.5 Summary and Conclusion

The validation of the MED-V-STEP framework by focus groups and questionnaires has demonstrated that the overall structure of the MED-V-STEP framework provides benefits in terms of:

- applying relevant international standards to address the quality and software testing related issues in the medical device domain,
- addressing software testing gaps in medical device domain-specific standard IEC 62304 with the use of generic, non-domain-specific standard ISO/IEC/IEEE 29119-2,
- providing the lower level of detail in terms of test processes for medical device software, and
- efficiency and low overhead of framework implementation due to integration and consolidation of the standards performed in this study.

The validation of the MED-V-STEP framework by focus groups and questionnaires has demonstrated that the relationships between test and development processes defined within the framework provide benefits in terms of:

- providing additional test clauses of ISO/IEC/IEEE 29119-2 mapped to relevant development stages of IEC 62304.
- providing information about the ISO/IEC/IEEE 29119-2 activities that need to be performed adhering to the logical dependencies with development clauses, and
- aligning ISO/IEC/IEEE 29119-2 test clauses with related IEC 62304 clauses to allow testing and development to work together and the synchronisation between the two.

The analysis of the quantitative data obtained from questionnaires revealed that the Software Testing Organisation from *Focus Group 2* rated the framework efficiency higher (5) compared with the slightly different rating (3.8) of the Medical Device Software Development Organisation from *Focus Group 1*. From a certain perspective, this would confirm the tendency of software development organisations to underestimate to some extent software testing compared to software testing organisations. From another perspective, however, it could be perceived that the testing organization does not have a thorough knowledge of IEC 62304 and therefore of what is required in relation to this standard. It has to be noted that there were different numbers of participants in focus groups, so their ratings cannot be directly evaluated or compared. However, two

participants from *Focus Group 2* were consistent and both rated the framework efficiency at 5 out of 5. In addition, the participants from both focus groups were consistent in qualitative evaluation and agreed that the framework is efficient in addressing testing gap coming from the IEC 62304 standard by providing information on the implementation of software testing best practice for medical device software. The high quantitative rating of both organizations therefore confirms the validity of the framework efficiency as presented in the findings above.

Focus groups also provided suggestions for possible improvements to the MED-V-STEP framework that have been identified as areas for future work on the framework. The MED-V-STEP framework, developed in this study in Excel file, can be adapted to the needs of a particular company, for example by developing a framework in the form of a searchable database, if it is required by the user. These suggestions for improvement are therefore included in section 6.5, where they are described as the subject of further work and research on the development of the framework.

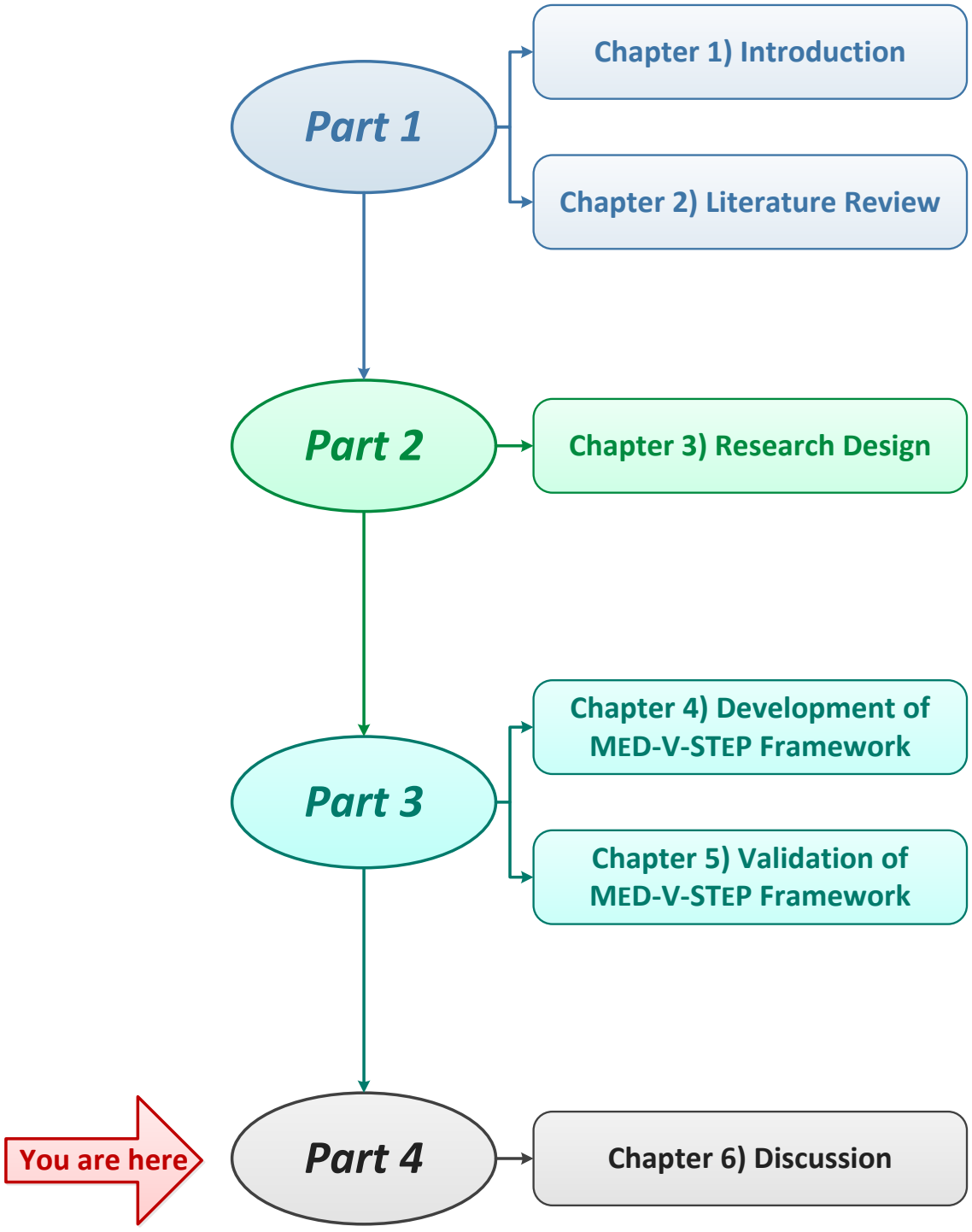
From the development of the MED-V-STEP framework, some design principles arise that could be applied to the development of other similar artefacts, as described below. The first lesson learned from the development of the MED-V-STEP framework is as follows. The previously established approaches, presented in Section 2.6, to integrate and consolidate standards, where one standard contains information that can address a gap coming from another standard, could be used for the development of an innovative software testing best practice framework for medical device software. In this study, the gap in the detailed information on software test activities, coming IEC 62304, was addressed using required software test processes from ISO/IEC/IEEE 29119-2.

The next lesson learned is as follows. The previously established approaches to mapping and defining relationships of multiple standards in order to achieve high efficiency of their joint implementation, could be used to align software testing best practice with the medical device software development life-cycle. ISO/IEC/IEEE 29119-2 generic software testing has previously been mapped and aligned with ISO/IEC 12207 generic software life-cycle processes, as shown in the annexed Table B.2 in ISO/IEC/IEEE 29119-2. Similarly, IEC 62304 medical device software life-cycle processes have been mapped to ISO/IEC 12207 generic software life-cycle processes, as shown in the annexed Table C.5 in IEC 62304. In this study, the mapping approach of standards community was leveraged, to map the ISO/IEC/IEEE 29119-2 test clauses and align with the IEC 62304 development clauses through the consolidation of these standards. The use and

combination of the two published mappings made by the standard community allowed the adoption of generic software testing best practice of ISO/IEC/IEEE 29119-2 and introducing it through IEC 62304 into the medical device domain via the ISO/IEC 12207 mapping. The use of the annexed tables ensures the validity of the detailed mapping performed in this study, and that the implementation of the MED-V-STEP framework will not affect compliance with IEC 62304.

Based on the lessons learned from the application of design principles for the purpose of this study, the development of the MED-V-STEP framework can serve as an example to develop software quality assurance best practice (MED-SQA) framework for medical device software. The MED-V-STEP framework can be extended with processes of other standards, such as Verification & Validation, Software Quality and Quality Management Systems standards, identified by the standards review in Section 2.4. Defining relationships between these standards will increase the efficiency of their implementation. The development of the MED-SQA framework represents another opportunity for future study. Chapter 4 revisits the research questions and objectives formulated to drive the research, presents research contributions made by the development and validation of MED-V-STEP framework, outlines research limitations and highlights recommendations for future research. This chapter ends with the conclusion of this study.

Map of Thesis – Part 4



Chapter 6 Discussion

6.1 Introduction

This thesis addresses existing quality and safety issues of medical device software, as evidenced by considerable adverse events for patients, such as injury or death, and the recalls of malfunctioning medical devices due to software failures. The aim of this thesis was to investigate how to prevent medical device software failures by improving software testing efficiency and whether developing a software testing best practice framework for medical device software can help to address the challenges of industrial software testing. A novel medical device verification software test process (MED-V-STEP) framework has been developed and validated to address the challenges related to testing medical device software.

At the beginning of this thesis, research questions were formulated along with corresponding research objectives to address these questions. These questions and objectives are revisited in the Section 6.2 with the analysis of the findings. Section 6.3 summarizes the contributions made through performing a literature review, development of the MED-V-STEP framework and validation of this framework. Next, Section 6.4 presents the research limitations of this study. In Section 6.5 it is recommended for future work to extend the developed MED-V-STEP framework to develop the MED-SQA framework. The final conclusions drawn from this study are presented in the last Section 6.6.

6.2 Research Questions and Research Objectives Revisited

In this thesis, the MED-V-STEP framework was developed to answer the overall Research Question:

How can generic software testing best practice be implemented to take into account the requirements of medical device software life-cycle processes related to software testing in order to address the software testing challenges in the medical device domain?

This Research Question has been divided into four Research Sub-Questions (RSQs) which have been addressed through the corresponding Research Objectives (RO). The following sub-sections discuss how the ROs addressed the identified RSQs, and the answers that were obtained.

6.2.1 Identifying Software Testing Challenges

RSQ.1 "What are the challenges associated with the implementation of generic and medical device software testing?"

By undertaking an extensive literature review, this was addressed through:

RO.1 "Investigate the challenges associated with the implementation of generic and medical device software testing."

The challenges that software development organisations face when implementing the testing of both generic and medical device software are associated with changes in generic and medical device software systems that are becoming larger in size and more complex due to the increasing functionality provided by the software. As a consequence of this, organisations must consider testing of increasing functionality and detection of defects in increasingly complex software systems.

According to the literature review, although software defects pose a threat to software functioning correctly, the software industry tends to prioritise programming over testing. Since the choice of method and scope of testing is the responsibility of the software development organisation, testing is often underestimated. This can be seen in non-safety critical software industry, where the testing effort accounts for 20% of time and costs within the whole software development life-cycle. This is much less than in safety critical domains, where testing effort accounts for 50% or more. As a result, development processes preferred by organisations were evolving to meet the requirement for the development of increasing functionality of software. However, software testing was not evolving fast enough to deal with these challenges effectively. This is evidenced by quality issues that increase with increasing size and complexity of software, and indicates the need for more extensive testing

Compared to generic software, testing in the medical device software industry is the most time consuming and expensive stage, representing 50% of all software development life-cycle stages, so there is no underestimation of testing here. This is due to the need to ensure safety of software, because defects in software pose a threat not only to the functionality of software, but the consequence of software malfunction may be the loss of human health or life. For this reason, medical device software is also subject to regulatory oversight. It is interesting that, despite the great effort on testing carried out in accordance with software life-cycle processes that meet regulatory requirements, the

challenges of testing the increasing functionality and detecting defects in increasingly complex software systems have not been fully addressed.

In medical device software development, it is common practice that some aspects of the development, such as quality managemenent or risk management, are not part of IEC 62304 and are expected to be addressed by the requirements of other standards, which are assumed to have already been implemented. This applies to sets of activities of ISO 14971 for risk management or ISO 13485 for quality management systems that are not defined in IEC 62304, but their fulfilment is mandatory for compliance with IEC 62304. As the necessary requirements for a given aspect have been introduced from the implemented standards, the IEC 62304 standard leverages these relevant sets of requirements while preventing their duplication or redefinition.

The existing practice of leveraging the required standards indicates the possibility of the use of other standards that are not required for compliance, but which contain best practice in a given aspect. This thesis has shown that organizations with an implemented IEC 62304 life-cycle face challenges related to medical device software testing, and considered if another software testing standard could be used along with the IEC 62304 standard to address these challenges. The medical device software industry could leverage software testing best practice as represented by ISO/IEC/IEEE 29119-2. It is worth recalling here that the IEC 62304 software life-cycle processes ensuring compliance with regulations do not specify in detail the test activities, which remain the responsibility of the medical device software development organization. As a result, the evidence of continuing and increasing adverse events due to malfunction of software indicates the need for software testing evolution in the medical device industry.

Given that complete testing of all quality aspects of increasingly complex software systems seems unrealistic due to limited time and human resources, the ISO/IEC/IEEE 29119 standard addresses this issue by providing a structured approach to well-designed software testing. This is based on the multi-layer test process model with test activities grouped into three process groups. For the test design and implementation process, Part 4 of this standard describes three sets of test design techniques, the selection of which depends on the application domain as well as the risk and criticality of the software. Part 4 also provides detailed guidance on how to implement the test design and implementation process activities for each technique.

A structured approach to well-designed software testing could be approach to deal with the increasing complexity of software and address generic software testing challenges

that also exisit in the medical device software domain. The need for the evolution of testing of medical device software in terms of effectiveness in detecting defects and implementation efficiency motivated the search for solutions to answer RSQ.2. The specification and review of RSQ.2 and RO.2 is presented in the following section.

6.2.2 Addressing Identified Challenges

RSQ.2 "How can the challenges associated with the implementation of software testing in the medical device domain be addressed?"

By performing a literature and standards review this has been addressed through:

RO.2 "Determine techniques that can be used to address the challenges associated with the implementation of medical device software testing."

RO.2 was formulated as a result of findings from RO.1, that there are challenges related to the testing of generic software and medical device software that need to be addressed to improve testing efficiency. To this end, international standardisation organisations have been publishing international standards related to generic software testing and medical device software development. The international standards section of the literature review shows that these standards have played and continue to play a significant role in the progression of software development and testing. In the case of harmonised standards, they are used as a benchmark to demonstrate compliance with regulatory requirements.

There are sets of international standards related to generic software testing, verification & validation and medical device software development. In addressing RO.2, international standards were reviewed to determine their relevance and limitations in addressing identified challenges as under RO.1. This review has demonstrated that the challenges of medical device software testing can be addressed with the implementation of relevant standards. At the same time, it was found that the ISO/IEC/IEEE 29119-2, IEC 62304 and ISO 14971 standards should be used to address the medical device software testing challenges. Since the development of medical device software requires the use of IEC 62304 which, however, does not provide detailed guidance on software testing, and such a detailed description of testing is contained in ISO/IEC/IEEE 29119-2, therefore generic software testing best practice of this standard can be used to address the software testing gap in the IEC 62304 standard to improve medical device software testing practice.

Generic software testing standard ISO/IEC/IEEE 29119-2 provides sets of test activities that improve testing efficiency, but it does not address how they can be adapted for use in the medical device software industry. Standards related to medical device software

address safety and compliance but do not provide detailed information on software test activities. Without a consolidated resource, information on medical device software testing has to be identified and collected from relevant standards and the relationships between these standards should be defined. Therefore, it difficult for organisations to efficiently leverage existing generic software testing practice for testing medical device software. Considering the fact that the implementation of one standard poses a significant challenge for many organisations, the need to implement multiple standards in a consolidated way would pose an even greater challenge. Thus the question becomes: how can this be implemented efficiently?

These findings enabled the definition of the research problem in this thesis as the distribution of the required information in multiple standards and the lack of this information in the form of a single consolidated standard. The identified research problem highlighted the need for integration and consolidation of software testing related information from multiple standards. The design and development of a software testing best practice framework for medical device software was proposed to address the identified research problem. Development of this framework addressing medical device software testing challenges identified as under RO.1 uses the information on software testing best practice of the ISO/IEC/IEEE 29119-2 standard to address the testing gap in the IEC 62304 medical device software development standard.

6.2.3 Developing a Software Testing Best Practice Framework

RSQ.3 "Can a framework be developed to implement software testing best practice while addressing the requirements of medical device software life-cycle processes related to software testing?"

This has been answered by the design and development of a software testing best practice framework for medical device software to address:

RO.3 "Develop the framework which incorporates software testing best practice while addressing the requirements of medical device software life-cycle processes related to software testing"

In response to RO.3, this thesis describes the design and development of the MED-V-STEP framework using processes of standards (RO.2) that contribute to addressing the challenges of testing medical device software (RO.1). The following two important issues were taken into account when designing the MED-V-STEP framework:

1. The targeted recipients of the MED-V-STEP framework are organisations that have implemented software development life-cycle in accordance with IEC 62304 and ISO 14971 as evidence of compliance with medical device regulatory requirements. It is worth recalling that regulatory compliance is mandatory for the introduction of medical device software on the market. For the development of the software testing framework in this study, this means that the use of the MED-V-STEP framework must in no way compromise the compliance of software development life-cycle with the requirements of IEC 62304 and ISO 14971.
2. Some IEC 62304 clauses deal with software testing, but do not provide a detailed description of the test activities. They remain the responsibility of the medical device software organisations, and therefore we do not know what test activities are used in the medical device software industry. This implied the requirement to apply current software testing best practice to the framework in order to improve industrial software testing practice.

From the above, it follows that the implementation of ISO 29119-2 software testing best practice to improve the effectiveness of industrial medical device software testing cannot be done in isolation from IEC 62304 and ISO 14971 but needs to take into account the requirements of these standards regarding software testing and risk management. IEC 62304 and ISO/IEC/IEEE 29119-2 define one of the basic principles of their effective application, consisting of applying related activities in accordance with their logical dependencies. Therefore, to use the best practice test model defined by ISO/IEC/IEEE 29119-2, the structure as well as interrelated processes and activities of this model had to be retained in the MED-V-STEP framework.

To ensure that this test model was applied in relation to IEC 62304 software life-cycle and ISO 14971 risk management processes, it was necessary to determine how they are related. The existence of relationships between these standards and the need to comply with them is demonstrated by the fact that ISO/IEC/IEEE 29119-2 in the annexed table defines relationships with ISO/IEC 12207. ISO/IEC 12207 similar to IEC 62304 defines software life-cycle processes, which, however, are not adapted to the needs of a particular industry but are intended for generic use. Based on ISO/IEC 12207, IEC 62304 was created, and their similarity is demonstrated in the annexed table in IEC 62304 mapping most of these standards' clauses.

Defined relationships between ISO/IEC/IEEE 29119-2 and ISO/IEC 12207, and mapped clauses of ISO/IEC 12207 to IEC 62304 indicated that there are relationships between

ISO/IEC/IEEE 29119-2 and IEC 62304 which have not been identified or specified prior to this study. The definition of the relationships between ISO/IEC/IEEE 29119-2, IEC 62304 and ISO 14971 was made in this study by mapping the above-mentioned annexed tables. The aim of this mapping was to identify existing relationships between test, development and risk management activities and define logical dependencies of related activities. As a result, all test processes of ISO/IEC/IEEE 29119-2 and as identified in this study, the related IEC 62304 and ISO 14972 activities were introduced into the software testing framework in accordance to their logical dependencies.

Since the related test and development activities were derived on the basis of annexed tables in ISO/IEC/IEEE 29119-2 and IEC 62304, the identified relationships maintain compliance with processes of these standards. Defining these relationships in the MED-V-STEP framework ensures that the implementation of the framework does not compromise the compliance of software development life-cycle with IEC 62304.

6.2.4 Validating MED-V-STEP Framework

RSQ.4 "Can the use of the framework contribute to the implementation of software testing best practice for the medical device software?"

RSQ.4 was formulated to examine to what degree the developed MED-V-STEP framework has the potential to address software testing challenges faced by medical device software development organisations. RSQ.4 has been addressed by conducting industrial validation as under RO.4.

RO.4 "Validate the framework in terms of providing information on the implementation of software testing best practice for the medical device software."

A qualitative approach in the form of a focus group and a quantitative approach in the form of a questionnaire were selected to address this objective and anwer RSQ.4. The overall structure of the MED-V-STEP framework and its efficiency in the future implementation were evaluated by industry participants from a Medical Device Software Development Organisation and a Software Testing Organisation.

Since the targeted recipients of the MED-V-STEP framework are organisations that develop medical device software, the primary focus of validating the MED-V-STEP framework was the evaluation made by a Medical Device Software Development Organisation. Participants in positions at various software development life-cycle stages, such as senior design quality assurance engineer or senior software developer, evaluated the MED-V-STEP framework as efficient:

- in enhancing requirements of IEC 62304 related to software testing with software testing best practice of ISO/IEC/IEEE 29119-2,
- in defining logical dependencies of related software test, development and risk management clauses, and
- in terms of implementing test activities in relation to development activities with as little overhead as possible.

This qualitative evaluation was complemented by a quantitative evaluation of the efficiency of the framework in the above areas, averaging 3.8 in the range of 1 to 5.

Another focus of validating the MED-V-STEP framework was the evaluation from testing and SQA perspective made by Software Testing Organisation. Senior SQA consultants with experience in software testing in safety-critical domains, knowledge of ISO/IEC/IEEE 29119-2 and knowledge about the integration of domain-specific requirements evaluated the MED-V-STEP framework as efficient:

- in providing needed test processes and determining their relationships with the medical device software development life-cycle and
- in providing lower level detail in terms of the test processes. They evaluated that providing such detail and granularity in terms of the test processes is useful.

They complemented their qualitative evaluation by a quantitative evaluation of the efficiency of the framework in the above areas, averaging 5 in the range of 1 to 5.

Feedback from focus groups and questionnaires confirmed the efficiency of the MED-V-STEP framework in:

- adapting software testing best practice of ISO/IEC/IEEE 29119-2 to the needs of organisations that have implemented and follow software life-cycle processes according to IEC 62304 and
- providing the relationships and logical dependencies of ISO/IEC/IEEE 29119-2 test clauses with IEC 62304 development clauses and ISO 14971 risk management clauses.

Participants identified the following potential benefit of the MED-V-STEP framework in addressing the software testing challenges in the medical device domain, stating that:

- the framework is efficient in improving medical device software testing with a low overhead of the framework implementation.

6.2.5 Addressing Overall Research Question

Previous sections discussed how ROs addressed the specific RSQs. This section summarizes how these ROs and RSQs contributed to answering the overall research question:

"How can generic software testing best practice be implemented to take into account the requirements of medical device software life-cycle processes related to software testing in order to address the software testing challenges in the medical device domain?"

To answer this overall research question, the challenges in testing medical device software were investigated, international standards contributing to addressing these challenges were reviewed, an innovative software testing framework for medical device software was developed based on the findings of literature and standards review, and the MED-V-STEP framework was validated for its impact on improving medical device software testing practice.

While this study revealed the insufficient evolution of software testing to deal with existing challenges, international standards contributing to addressing this insufficient evolution were identified. Since the solution to the software testing challenges was based on these standards, another challenge has been revealed associated with the implementation of multiple standards in a consolidated way. The MED-V-STEP framework was designed and developed to address this challenge by providing a consolidated set of activities for medical device software testing that can be implemented in an efficient way.

To evaluate an innovative software testing approach for medical device software that can be used in the future, the MED-V-STEP framework has been validated by industry organizations by obtaining feedback on their perception of the framework efficiency. The industrial evaluation confirmed the validity of the framework in improving the efficiency of medical device software testing. The validation also identified future work and concluded with the requirement for the future implementation of the MED-V-STEP framework and the development of the MED-SQA framework. The contributions of this research are presented in Section 6.3.

6.3 Research Contributions

Section 1.7 outlined the key contributions to be made in relation to the development of a software testing best practice framework for medical device software. This section summarizes how these contributions were met by this study.

The first contribution in Section 2.2 was identification of challenges that need to be considered when improving the efficiency of generic software testing and demonstration of the resulting software quality issues. Section 2.3 has contributed to identifying the challenges to be considered when testing medical device software and demonstrated the resulting safety issues of medical device software.

The next contribution was provided in Section 2.4 with a comprehensive overview of international standards on generic software testing and software quality as well as standards helping medical device software development organisations to comply with medical device regulations. The review of standards highlighted the relevance and limitations of standards in addressing the identified challenges associated with medical device software testing. The findings of this standards review focussed the research problem and provided the basis for proposing a solution to the research problem in this thesis.

Another key contribution was provided in Chapter 4 by developing an innovative MED-V-STEP framework. The mapping of ISO/IEC/IEEE 29119-2, IEC 62304 and ISO 14971 contributes to the knowledge of adapting the ISO/IEC/IEEE 29119-2 software testing model to the requirements in the medical device domain, which was not performed prior to this study. The framework structure based on ISO/IEC/IEEE 29119-2 software testing model helps medical device software development organisations benefit from implementing best practice software testing.

One of the key contributions was to define in the MED-V-STEP framework relationships between test, development and risk management activities that provide information on how to extend IEC 62304 and ISO 14971 requirements regarding software testing with sets of ISO/IEC/IEEE 29119-2 software test activities. The defined relationships help organisations to benefit from implementing software testing best practice of ISO/IEC/IEEE 29119-2 while addressing the requirements of IEC 62304 and ISO 14971 regarding software testing.

A key contribution was also to define the logical dependencies between related clauses that inform which activity generates output as input to enhance related activity. Logical

dependencies help organizations benefit from the fact that test activities are enhanced by the output of related development activity, and development activities are enhanced by the output of related test activity.

The next key contribution was provided by demonstrating that the MED-V-STEP framework improves software testing industrial practice with a low overhead of the framework implementation. The validation presented in Chapter 5 contributed to the knowledge about the efficiency of the framework in future industrial implementation and the associated benefits. This contribution is limited in that the framework has not been implemented in this study.

The last key contribution was to provide the lesson learned from the design and development of the MED-V-STEP framework for the future extension of the framework with other quality and verification & validation clauses dispersed between standards identified and presented in Chapter 2. This is provided in Section 6.5 by outlining future development of a software quality assurance best practice (MED -SQA) framework for medical device software. The aim of the MED -SQA framework development is to help medical device software organisations to implement verification & validation efficiently in order to achieve good quality and safe software. This study generated several key contributions presented above, while the next section outlines the research limitations in this study.

6.4 Research Limitations

The review of international standards identified set of four standards related to generic software testing, verification & validation and quality and another set of four standards related to medical device software development. The information on the efficient application of software testing and verification & validation of medical device software is distributed in all eight standards. They address various aspects of SQA and software testing challenges and are relevant to contribute to addressing these challenges.

A limitation of this study is that the MED-V-STEP framework development focussed on some and not all standards identified in Section 2.4.2 and presented in Figure 2. A strategic decision to perform the mapping of three standards was made for the following reasons:

- Medical device software testing is critical to software quality and safety. However, the evolution of software testing industrial practice was insufficient to deal with software testing challenges. For this reason, the identified software testing challenges need to be addressed first, taking into account the software quality and safety issues demonstrated in this study.
- The MED-V-STEP framework is intended for organisations that develop software to be embedded or an integral part of medical device, in accordance with the IEC 62304 software life-cycle processes to ensure compliance with medical device regulations. Since IEC 62304 requires the verification to be performed for each life-cycle stage, and the requirements of this standard for software testing within verification need to be fulfilled, this study focused on implementing software testing best practice in relation to verifying medical device software. The MED-V-STEP framework is intended for testing medical device software and does not cover validation of the final medical device. Therefore IEC 82304-1 and IEC 60601-1 standards needed at the system level for validation and final release of the medical device, even when the medical device consists entirely of software, are not included in the MED-V-STEP framework. Consequently, not all ten identified standards that contribute to improving the quality of software, but only those three standards contributing to software testing within verification of medical device software were used to develop the MED-V-STEP framework.

This limitation creates an opportunity for further research, as outlined in Section 6.5, which describes what can be done in future work, and is addressed by proving the

approach to developing best practice MED-SQA framework for medical device software with the use of all identified standards.

Another limitation of this study is that the MED-V-STEP framework has not been implemented. Validation of the framework was based on how industry organisations perceive the efficiency of the framework in improving medical device software testing practice in future implementation rather than in actual implementation. To mitigate this limitation, validation was performed with two organisations, a Medical Device Software Development Organisation and a Software Testing Organisation, to obtain feedback from these two different perspectives. However, an assessment of the structure by other medical device software organizations would be beneficial for validating the usefulness of the framework for improving the quality and safety of the medical device software. Additionally, prior to the validation, the development approach and progressing framework development were presented at the international 11th Systems Testing and Validation Workshop and to the Software Testing Organisation. However, an assessment of the structure by other medical device software organizations would be beneficial for validating the usefulness of the framework for improving the quality and safety of the medical device software. The discussed limitations of this study form the basis for future work which is discussed in the following section.

6.5 Future Work

This section outlines directions for future research. Focus groups participants' suggestions for improving the MED-V-STEP framework mainly expressed requirements for the usability of the framework for their purpose. Since the aim of this study was to develop the innovative software testing practice that addresses the research problem in general rather than in a particular organisation, these suggestions indicate opportunities for future work and would be subject to the further development of the framework. Ongoing development of the software testing framework should include an extensive opportunity for expert and practitioner input.

Since the framework was developed by mapping existing standards, and the relationships and logical dependencies between the standards were defined with the help of high-level tables in the standards appendices, the standard community would be competent to validate the development activities performed in this study and the MED-V-STEP framework itself. A standards community review would evaluate the relevance of mapping, the definition of logical dependencies, and the structure of the framework.

As the framework is intended for use by organisations developing medical device software compliant with IEC 62304, and dealing with challenges associated with software quality and testing, such organisations would be suitable subjects to validate the efficiency of the framework implementation in addressing the software testing challenges. The framework creates opportunities for its implementation as a database tailored to the needs of a particular company. The industrial pilot implementation would evaluate the feasibility of implementing the framework and its efficiency in providing information on how activities need to be performed to maintain the defined logical dependencies.

Other opportunities for the future work include the extension of the developed MED-V-STEP framework with the use of other standards described in the standards review section. The MED-V-STEP framework could be extended to develop a software quality assurance best practice (MED-SQA) framework for medical device software. Figure 13 presents a proposal of a MED-SQA framework consisting of SQA clauses of the standards presented in Section 2.4.3 and Section 2.4.4. These standards have been identified as relevant to address various aspects of identified SQA challenges. They are related to the areas presented by dotted rectangles, such as generic software testing, software quality, verification & validation, medical device software development, risk management and quality management systems.

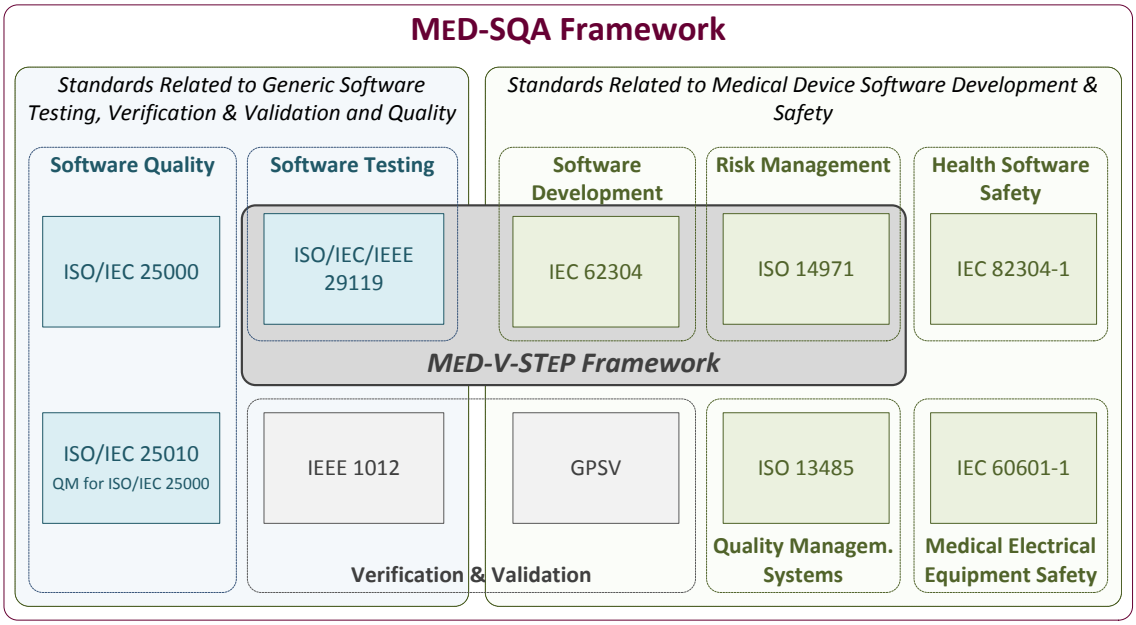


Figure 13 MED-SQA Framework

The future development and validation of the MED-SQA framework would consist of the following steps:

Incorporating Standards into the MED-SQA framework

- The MED-V-STEP framework will incorporate IEEE 1012 verification & validation activities that are not related to software testing.
    - The ISO/IEC/IEEE 29119-2 test activities will be aligned with IEEE 1012 sets of activities such as analysis, reviews, inspections and others, for each software development stage of IEC 62304.
    - These sets of activities will enhance the framework efficiency in an assessment of medical device software development processes throughout the software development life-cycle. Examples are the assessment of correctness, completeness, consistency, as well as testability of requirements or conformity of the activity output with a given requirements of that activity.
- The MED-V-STEP framework will be extended with the GPSV and ISO 13485 clauses to include a set of validation activities applicable to the development of medical device software.
    - This set of validation activities will enhance the framework efficiency in the assessment, whether the software satisfies its intended use and user needs.

- The extension of the framework will include other SQA related clauses of the software quality standards ISO/IEC 25000 and ISO/IEC 25010 to cover requirements specification and evaluation of software quality.
    - This extension will enhance the framework efficiency in aligning customer definitions of quality with attributes of the development process of complex and extensive software systems.
- The MED-V-STEP framework will include requirements of the standards IEC 82304-1 and IEC 60601-1 to cover the safety of healthcare and programmable electrical medical systems.
    - This extension will enhance the framework efficiency in providing safety requirements for software which provides safety functionality for programmable electrical medical systems, as well as for healthcare software.

Defining relationships between Incorporated Standards

- Based on the experience from the development of the MED-V-STEP framework, the related verification & validation, quality and medical device SQA related clauses could be mapped and their logical dependencies defined.

Validating the MED-SQA framework

- The potential validation method of the proposed MED-SQA framework could involve a standards community review with the focus on the relevance and completeness of incorporated clauses and their relationships.
- Also, the MED-SQA framework could be refined by pilot implementation in a medical device software development organisation with the aim to assess the MED-SQA framework efficiency in improving SQA practices.

6.6 Conclusion

A number of conclusions in relation to the MED-V-STEP framework have been drawn in this study and are presented in this section. The current trend for medical device software systems to provide more and more functionality indicates that software safety and quality issues and related software testing challenges that already exist will increase. Delay in addressing these challenges will result in a growing gap in the evolution of software testing that is becoming increasingly difficult to deal with.

The medical device software industry requires improvements in software testing to address an increasing number of medical device malfunctions due to defective software, resulting in patient's injury or loss of life. Adverse events and safety issues caused by defective software have made it of great importance to provide safe software for medical devices. This importance is evidenced by national and international regulatory oversight of medical device software development and the publication of standards harmonised with these regulations, in order to improve the quality and safety of software. Since, despite the effort of regulatory bodies, standardisation organisations and software development organisations performing extensive software testing compliant with medical device regulations, there is still increasing trend of adverse events and safety issues due to software containing defects, it can be therefore concluded that there is a significant challenge with ensuring software safety and this thesis and the development of the framework takes a small step to begin to address the identified challenges.

The organisations who validated the framework remarked on the benefits of testing using standards' software testing best practice and using the MED-V-STEP framework rather than multiple standards. The framework was validated as improving the software testing efficiency by defining the relationships and logical dependencies between the test, development and risk management activities that are integrated and consolidated into the framework, thus enabling the efficient implementation of software testing best practice for medical device software. Since organisations can implement sets of necessary activities without having to identify and collect them from multiple standards and without having to define the related activities where testing interacts with development and risk management activities, they can implement them with "little overhead" as stated during the validation process. Hence it can be concluded that the MED-V-STEP framework is beneficial for organisations in terms of implementation efficiency with low implementation overhead.

The validation also proved, through the development of the MED-V-STEP Framework, that this approach could be used in the development of a larger MED-SQA framework for medical device software as part of future research in this area. Since the development approach of the framework was validated, it can be therefore concluded that future enhancement of the framework with other verification & validation and software quality standards can be performed and the MED-SQA framework can be developed with this development approach.

References

Alemzadeh, H., Iyer, R.K., Kalbarczyk, Z. and Raman, J. (2013). Analysis of safety-critical computer failures in medical devices. *IEEE Security & Privacy*, *11*(4), pp. 14-26.

Avgeriou, P., Kruchten, P., Nord, R.L., Ozkaya, I. and Seaman, C. (2015). Reducing friction in software development. *IEEE Software*, *33*(1), pp. 66-73.

Bertolino, A. (2007). Software testing research: Achievements, challenges, dreams. In *Future of Software Engineering (FOSE'07)* (pp. 85-103). IEEE.

Bujok, A.B., MacMahon, S.T., Grant, P., Whelan, D., Rickard, W.J., and McCaffery, F. (2017). Approach to the development of a unified framework for safety critical software development. *Computer Standards & Interfaces*, *54*, pp. 152-161. Available from: https://www.sciencedirect.com/science/article/abs/pii/S0920548916301921 [accessed: April 17th 2020].

Camara, C., Peris-Lopez, P. and Tapiador, J.E. (2015). Security and privacy issues in implantable medical devices: A comprehensive survey. *Journal of Biomedical Informatics*, 55, pp. 272–289. Available from: https://www.sciencedirect.com/science/article/pii/S153204641500074X?via%3Dihub [accessed: April 17th 2020].

Charette, R.N. (2009). This car runs on code. *IEEE Spectrum*, *46*(3), p. 3.

Chillarege, R. (1999). Software Testing Best Practices. IBM Thomas J. Watson Research Division.

Clancy, T. (2014). The standish group chaos report. *Project Smart*, pp. 8-9.

Clarke, P. and O'Connor, R. (2010). Harnessing ISO/IEC 12207 to Examine the Extent of SPI Activity in an Organisation. In *European Conference on Software Process Improvement* (pp. 25–36) Springer, Berlin, Heidelberg.

Creswell, J.W. (2016). Advances in Mixed Methods. *Webinar – Mixed Methods International Research Association*, pp.1–65. Available from: https://cloudfront.ualberta.ca/-/media/ualberta/faculties-and-programs/centres-institutes/international-institute-of-qualitative-methods/webinars/mixed-methods/2016/jcreswellmmira-webinar.pdf [accessed: April 17th 2020].

Crotty, M. (1998). *The foundations of social research: Meaning and perspective in the research process*. Sage.

Dawson, C. (2009). Introduction to Research Methods. A practical guide for anyone undertaking a research project, How to Content.

Dix, J.R., Ramachandran, S. and Oppenheimer, D.S. (2016). What's Behind Medtech's Recall Epidemic*? Medical Device and Diagnostic Industry News Products and Suppliers*

DKIT. (2015). DKIT Research Ethics Policy. Available from: https://www.dkit.ie/assets/uploads/documents/Research/Policies/DkIT%20Research%20Ethics%20Policy.pdf [accessed: April 17th 2020].

European Commission. *Harmonised standards for medical devices*. [Online]. Available from: https://eur-lex.europa.eu/eli/dec_impl/2020/437/oj [accessed: April 17th 2020].

European Commission. (2016). Guidance document Medical Devices-Scope, field of application, definition-Qualification and classification of stand alone software-MEDDEV 2.1/6.

European Union. (2017). Legislation L 117. *Official Journal of the European Union*, 60. Available from: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ:L:2017:117:FULL&from=EN [accessed: April 17th 2020].

Feast, L. and Melles, G. (2010). Epistemological positions in design research: A brief review of the literature. In *2nd International Conference on Design Education, University of New South Wales, Sydney, Australia.*

Fernández, D.M., Wagner, S., Lochmann, K., Baumann, A. and de Carne, H. (2012). Field study on requirements engineering: Investigation of artefacts, project parameters, and execution strategies. *Information and Software Technology*, 54(2), pp.162–178.

Food and Drug Administration. *FDA Agents: FDA Registration and U.S. Agent Representation.* [Online]. Available from: http://www.fdaagents.com/ [accessed: April 17th 2020].

Food and Drug Administration. (2002). General principles of software validation; final guidance for industry and FDA staff. *Food and Drug Administration, 11*.

Food and Drug Administration. (2018). Quality system (QS) regulation/medical device good manufacturing practices. Available from: https://www.fda.gov/medical-devices/postmarket-requirements-devices/quality-system-qs-regulationmedical-device-good-manufacturing-practices [accessed: April 17th 2020].

Food and Drug Administration. (1990). Safe Medical Devices Act of 1990. *Public Law 101-629.* Available from: https://www.govinfo.gov/content/pkg/STATUTE-104/pdf/STATUTE-104-Pg4511.pdf#page=16 [accessed: April 17th 2020].

Food and Drug Administration. (1997). *The FDA and worldwide quality system requirements guidebook for medical devices*. Asq Press.

Gelperin, D. and Hetzel, B. (1988). The growth of software testing. *Communications of the ACM, 31*(6), pp. 687-695.

Guba, E.G. and Lincoln, Y.S. (1994). Competing paradigms in qualitative research. *Handbook of qualitative research*, *2*(163-194), p.105.

Graham, D., Van Veenendaal, E. and Evans, I. (2008). *Foundations of software testing: ISTQB certification.* Cengage Learning EMEA

Hailpern, B. and Santhanam, P. (2002). Software debugging, testing, and verification. *IBM Systems Journal, 41*(1), pp. 4-12.

Hastie, S. and Wojewoda, S. (2015). Standish group 2015 chaos report-q&a with jennifer lynch. *Retrieved, 1*(15), p. 2016. Available from: https://www.infoq.com/articles/standish-chaos-2015/ [accessed: April 17th 2020].

Heimdahl, M.P. (2007). Safety and software intensive systems: Challenges old and new. In *Future of Software Engineering (FOSE'07)* (pp. 137–152). IEEE.

Hevner, A.R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems*, 19(2), p. 4. Available from: https://aisel.aisnet.org/sjis/vol19/iss2/4/ [accessed: April 17th 2020].

Hevner, A.R., March, S.T., Park, J. and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, pp. 75–105.

IEC. *International Electrotechnical Commission - About the IEC.* [Online]. Available from: https://www.iec.ch/about/?ref=menu [accessed: April 17th 2020].

IEC. (2020). *SC 62A – Common aspects of electrical equipment used in medical practice* [Online]. Available from: https://www.iec.ch/dyn/www/f?p=103:22:28481321188541::::FSP_ORG_ID,FSP_LANG_ID:1359,25 [accessed: April 17th 2020].

IEC. (2015). *IEC 62304:2006+A1:2015 Medical device software - Software life-cycle processes.* Geneva, Switzerland, International Electrotechnical Commission.

IEC. (2016) *IEC 82304-1:2016 Health software General requirements for product safety.* Geneva, Switzerland, International Electrotechnical Commission.

IEC. (2012) *IEC 60601-1:2005+AMD1:2012 Medical electrical equipment – General requirements for basic safety and essential performance.* Geneva, Switzerland, International Electrotechnical Commission.

IEEE. *Institute of Electrical and Electronics Engineers - Mission & Vision.* [Online]. Available from: https://www.ieee.org/about/vision-mission.html [accessed: April 17th 2020].

IEEE. (2016). *IEEE 1012-2016 - Standard for System, Software, and Hardware Verification and Validation*. New York, USA, IEEE Computer Society.

Iivari, J. and Venable, J.R. (2009). Action research and design science research-seemingly similar but decisively dissimilar.

ISO. (2020a). *ISO/IEC JTC 1/SC 7 - Software and systems engineering* [Online]. Available from: https://www.iso.org/committee/45086.html [accessed: April 17th 2020].

ISO. (2020b). *STANDARDS BY ISO/TC 210 - Quality management and corresponding general aspects for medical devices.* [Online]. Available from: https://www.iso.org/committee/54892/x/catalogue/p/1/u/0/w/0/d/0 [accessed: April 17th 2020].

ISO. (2016). *ISO 13485:2016 Medical devices — Quality management systems — Requirements for regulatory purposes*. Geneva, Switzerland, International Organisation for Standardization.

ISO. (2012). *ISO 14971:2012 Medical devices — Application of risk management to medical devices*. Geneva, Switzerland, International Organisation for Standardization.

ISO. (2008). *Integrated use of management system standards*. Geneva, Switzerland, International Organisation for Standardization.

ISO. *the International Organization for Standardization - Strategy 2016-2020.* [Online]. Available from: https://www.iso.org/files/live/sites/isoorg/files/store/en/PUB100364.pdf [accessed: April 17th 2020].

ISO. *Technical Committees.* [Online]. Available from: https://www.iso.org/technical-committees.html [accessed: April 17th 2020].

ISO/IEC. (2014). *ISO/IEC 25000:2014 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE.* Geneva, Switzerland.

ISO/IEC. (2011). *ISO/IEC 25010:2011 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.* Geneva, Switzerland.

ISO/IEC. (1995). *ISO/IEC 12207:1995: Information technology - Software life-cycle processes.* Geneva, Switzerland.

ISO/IEC. (2008). *ISO/IEC 12207:2008 Systems and software engineering — Software life-cycle processes.* Geneva, Switzerland.

ISO/IEC/IEEE. (2013a). *ISO/IEC/IEEE 29119-1:2013 Software and systems engineering — Software testing Part 1 : Concepts and definitions.* Geneva, Switzerland.

ISO/IEC/IEEE. (2013b). *ISO/IEC/IEEE 29119-2:2013 Software and systems engineering — Software testing Part 2 : Test processes.* Geneva, Switzerland.

ISTQB. (2016). *Standard Glossary of Terms used in Software Testing Version 3.2*. Available from: https://www.istqb.org/downloads/glossary.html [accessed: April 17th 2020].

Jackson, D. (2009). A direct path to dependable software. *Communications of the ACM, 52*(4), pp. 78-88.

Järvinen, P. (2012). On boundaries between field experiment, action research and design research.

Kaner, C. (2003). Fundamental challenges in software testing. In *Colloquium at Butler University*.

Keogh, O. (2015). MDevSpice is a one-stop shop for medical-device manufacturers. *The Irish Time*. Available from: https://www.irishtimes.com/business/mdevspice-is-a-one-stop-shop-for-medical-device-manufacturers-1.2147413 [accessed: April 17th 2020].

King, R. (2015). *Colliding Cultures: Software Development and the Medical Device Industry.* [Online]. Available from: https://www.linkedin.com/pulse/colliding-cultures-software-development-medical-device-russ-king [accessed: April 17th 2020].

Knight, J.C. (2002). Safety critical systems: challenges and directions. In *Proceedings of the 24th international conference on software engineering* (pp. 547-550).

Langenfeld, V., Post, A. and Podelski, A. (2016). Requirements defects over a project lifetime: an empirical analysis of defect data from a 5-year automotive project at Bosch. In *International Working Conference on Requirements Engineering: Foundation for Software Quality* (pp. 145-160). Springer, Cham.

Lee, I., Pappas, G.J., Cleaveland, R., Hatcliff, J., Krogh, B.H., Lee, P., Rubin, H. and Sha, L. (2006). High-confidence medical device software and systems. *Computer, 39*(4), pp. 33–38.

Lee, I. and Sokolsky, O. (2010). Medical cyber physical systems. In *Design automation conference*, pp.743–748. IEEE

Leveson, N.G. (2000). System safety in computer-controlled automotive systems. *SAE transactions*, *109*, pp. 287-294.

Mersino, A. (2018). *Project Success Rates Agile vs Waterfall.* [Online]. Available from: https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/ [accessed: April 17th 2020].

Myers, G.J. (2004). *The art of software testing*. John Wiley & Sons.

Mc Hugh, M., Mc Caffery, F. and Casey, V. (2011a). How amendments to the Medical Device Directive affect the development of medical device software. Available from: https://eprints.dkit.ie/39/ [accessed: April 17th 2020]

Mc Hugh, M., McCaffery, F. and Casey, V. (2011b). US FDA releases final rule on Medical Device Data Systems-what does this mean for device manufacturers?. *Journal of Medical Device Regulations 8*(3), pp.35-40.

McHugh, M., McCaffery, F., MacMahon, S.T. and Finnegan, A. (2013). Improving Safety in medical devices from concept to retirement. In *Handbook of Medical and Healthcare Technologies* (pp. 453-480). Springer, New York, NY.

Meenakshi, D., Naik, J.S. and Reddy, M.R. (2014). Software testing techniques in software development life cycle. *International Journal of computer science and information Technologies (IJCSIT)*, 5, pp. 3729–3731.

Monti, M.M., Vanhaudenhuyse, A., Coleman, M.R., Boly, M., Pickard, J.D., Tshibanda, L., Owen, A.M. and Laureys, S. (2010). Willful modulation of brain activity in disorders of consciousness. *New England Journal of Medicine, 362*(7), pp. 579-589. Available from: https://www.nejm.org/doi/pdf/10.1056/NEJMoa0905370 [accessed: April 17th 2020].

Mujumdar, A., Masiwal, G. and Chawan, P.M. (2012). Analysis of various software process models. *International Journal of Engineering Research and Application (IJERA), 2*(3), pp.2015–2021. Available from: https://www.researchgate.net/profile/Pramila_Chawan/publication/316510707_Analysis_of_various_Software_Process_Models/links/54f0aa150cf2f9e34efd0776/Analysis-of-various-Software-Process-Models.pdf [accessed: April 17th 2020].

National Cancer Institute. (2019). *Radiation Therapy to Treat Cancer.* [Online]. Available from: https://www.cancer.gov/about-cancer/treatment/types/radiation-therapy#q1 [accessed: April 17th 2020].

Note, A.I. (2014). SOFTWARE QUALITY ASSURANCE. Available from: https://www.researchgate.net/profile/Dr_Qaim_Mehdi_Rizvi/publication/260427593_Software_Quality_Assurance_An_Introductory_Note/links/00b49531412046f141000000.pdf [accessed: April 17th 2020].

Oates, B.J. (2005). *Researching information systems and computing*. Sage.

Papert, S. and Harel, I. (1991). Situating constructionism. *Constructionism, 36*(2), pp. 1-11.

Pauli, D. (2016). Fatal flaws in ten pacemakers make for Denial-of-Life attacks. *Viitattu, 14*, p. 2017.

Peffers, K., Tuunanen, T., Rothenberger, M.A. and Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems, 24*(3), pp. 45–77.

Planning, S. (2002). The economic impacts of inadequate infrastructure for software testing. *National Institute of Standards and Technology*.

Rastgarpour, M. and Shanbehzadeh, J. (2011). Novel Classification of Current Methods, Available Softwares and Datasets in Medical Image Segmentation. In *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)* (p. 1). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Reid, S. (2013). *ISO/IEC/IEEE 29119: the new international software testing standards. Testing Solutions Group.*

Saunders, M.N.K and Tosey, P.C. (2013). The layers of research design. *Rapport*, (Winter), pp. 58–59.

Sein, M.K., Henfridsson, O., Purao, S., Rossi, M. and Lindgren, R. (2011). Action design research. *MIS quarterly*, pp. 37-56.

Selwood, D. (2012). *Software That Can Kill. Electronic Engineering Journal.* [Online]. Available from: https://www.eejournal.com/article/20120711-swkills/ [accessed: April 17th 2020].

Seth, F.P., Taipale, O. and Smolander, K. (2014). Organizational and customer related challenges of software testing: An empirical study in 11 software companies. In *2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS)*, (pp. 1-12). IEEE.

Simon, H.A. (1996). The sciences of the artificial. *Cambridge, Massachusetts: MIT Press. Humphrey, J. & Schmitz, H.(2001). Governance in global value chains. IDS Bulletin, 32*, pp.19-23.

Simone, L.K. (2013). Software-related recalls: an analysis of records. *Biomedical instrumentation & technology*, 47(6), pp. 514–522.

Sivakumar, M.S., Casey, V., McCaffery, F. and Coleman, G. (2011). Improving verification & validation in the medical device domain. In *European Conference on Software Process Improvement* (pp. 61-71). Springer, Berlin, Heidelberg.

SRCL. (2018). *Stericycle Expert Solutions Recall Index: Medical Device*. [Online]. Available from: https://www.stericycleexpertsolutions.com/wp-content/uploads/2018/05/Expert-Solutions-US-Recall-Index-Q1-2018.pdf?__hstc=51647990.615dee5444e9411b29f1bd4384808d58.1587325150475.1587325150475.1587325150475.1&__hssc=51647990.1.1587325150476&__hsfp=1902762998 [accessed: April 17th 2020].

Steinbrook, R. (2005). The controversy over Guidant's implantable defibrillators. *New England Journal of Medicine*, *353*(3), pp. 221-224.

Tellis, W. (1997). Introduction to case study. *The qualitative report*, 269.

Thornhill, A., Saunders, M. and Lewis, P. (2009). *Research methods for business students*. Prentice Hall: London

Tian, J. (2005). *Software quality engineering: testing, quality assurance, and quantifiable improvement*. John Wiley & Sons.

Toussaint, N., Souplet, J.C. and Fillard, P. (2007). MedINRIA: medical image navigation and research tool by INRIA.

Tremblay, M.C., Hevner, A.R. and Berndt, D.J. (2010). Focus groups for artifact refinement and evaluation in design research. *Cais*, *26*(27), pp. 599–618.

Turing, I.B.A. (1950). Computing machinery and intelligence-AM Turing. *Mind*, *59*(236), pp. 433.

U.S. Government Publishing Office. (2015). Electronic Code of Federal Regulations.

de la Vara, J.L., Borg, M., Wnuk, K. and Moonen, L. (2014). Survey on safety evidence change impact analysis in practice : Detailed description and analysis. *Simula Research Laboratory, Tech. Rep*.

Vector Software. (2016). 2016 Software Testing Technology Report.

Vogel, D.A. (2011). *Medical Device Software Verification, Validation and Compliance*. Artech House

Wallace, D.R. and Kuhn, D.R. (1999). Lessons from 342 medical device failures. In *Proceedings 4th IEEE International Symposium on High-Assurance Systems Engineering* (pp. 123–131). IEEE.

Wallace, D.R. and Kuhn, R.D. (2001). Failure modes in medical device software: an analysis of 15 years of recall data. *International Journal of Reliability, Quality and Safety Engineering*, *8*(04), pp. 351-371.

Whittaker, J.A. (2000). What Is Software Testing? And Why Is It So Hard?. *IEEE Software*, *17*(1), pp. 70-79.

Zelkowitz, M.V. (2013). Education of software engineers. In *Perspectives on the Future of Software Engineering* (pp. 349-358). Springer, Berlin, Heilderberg.

Appendix A: Mapping Table of ISO/IEC/IEEE 29119-2 and IEC 62304

Mapping Explanation	ISO/IEC/IEEE 29119-2 Test Clause	ISO/IEC 12207 Clause (Process/ Activity/ **Task**)	ISO/IEC 12207 Clause (Process/ Activity/ **Task**)	IEC 62304 Development Clause
This is supported by clause 6 Organizational Test Process, which supports the periodic review of organizational test documentation, which can include the definition of processes for supporting testing.	6 Organizational Test Process	6.2.1 Life-Cycle Model Management 6.2.1.3.2 Process assessment **6.2.1.3.2.2**	N/A	N/A
The definition of risk management processes for testing is supported by clause 6.2 Organizational Test Process, which can be used to define organizational processes for managing and documenting testing-related risks.	6 Organizational Test Process 6.2 Organizational Test Process	6.3.4 Risk Management (note: the mapping for this clause is only applicable for identification and Mitigation of risks that can be mitigated by testing) 6.3.4.3.1 Risk management planning **6.3.4.3.1.1** **6.3.4.3.1.2**	6.3.4 Risk Management	5.1.7 Software risk management planning
This is supported by clauses 6.2.4.1, 6.2.4.2 and 6.2.4.3 of the Organizational Test Process, which supports the monitoring, reviewing and updating of risk management processes that are used in testing.	6 Organizational Test Process 6.2 Organizational Test Process 6.2.4.1 Develop Organizational Test Specification (OT1), 6.2.4.2 Monitor and Control use of Organizational Test Specification (OT2), and 6.2.4.3 Update Organizational Test Specification (OT3)	6.3.4 Risk Management 6.3.4.3.1 Risk management planning **6.3.4.3.1.5**	6.3.4 Risk Management	5.1.7 Software risk management planning
Establishing a risk profile for is supported by clause 6 Organizational Test Process, which can be used to define processes, thresholds, and profiles for testing-related risks for a system, a project, a series or projects or an entire organization.	6 Organizational Test Process 6.2 Organizational Test Process	6.3.4 Risk Management 6.3.4.3.2 Risk profile management **6.3.4.3.2.1**	6.3.4 Risk Management	5.1.7 Software risk management planning
This is supported by clause 6.2.4.1 Develop Organizational Test Specification (OT1), which can include the definition of a risk profile for the organization.	6 Organizational Test Process 6.2 Organizational Test Process 6.2.4.1 Develop Organizational Test Specification (OT1)	6.3.4 Risk Management 6.3.4.3.2 Risk profile management **6.3.4.3.2.3**	6.3.4 Risk Management	5.1.7 Software risk management planning
This is supported by clause 6.2.4.1 Develop Organizational Test Specification (OT5), which can be used to review the effectiveness and efficiency of organizational testing processes defined in an organizational test specification.	6 Organizational Test Process 6.2 Organizational Test Process 6.2.4.1 Develop Organizational Test Specification (OT1)	6.3.4 Risk Management 6.3.4.3.6 Risk management process evaluation **6.3.4.3.6.2**	6.3.4 Risk Management	5.1.7 Software risk management planning

This is supported by clause 6.2.4.1 Develop Organizational Test Specification (OT1), which supports recording and agreement of measurement processes for testing across all projects.	6 Organizational Test Process 6.2 Organizational Test Process 6.2.4.1 Develop Organizational Test Specification (OT1)	6.3.7.3 Measurement 6.3.7.3.1 Measurement planning 6.3.7.3.1.6 **6.3.7.3.1.7**	N/A	N/A
This is supported by clause 7.2 Test Planning Process, which provides a generic process for planning testing, which can include the identification and documentation of testing requirements.	7 Test Management Processes 7.2 Test Planning Process	6.1.1.3 Acquisition 6.1.1.3.1 Acquisition preparation **6.1.1.3.1.2**	N/A	N/A
For the planning of testing-related work, this clause is supported by clause 7 Test Management Processes, which provides activities for supporting test planning and test strategy development.	7 Test Management Processes 7.2 Test Planning Process	6.1.2 Supply 6.1.2.3.4 Contract execution **6.1.2.3.4.5**	N/A	N/A
The planning of testing is supported by clause 7.2 Test Planning Process, which provides generic processes for planning any phase or type of testing.	7 Test Management Processes 7.2 Test Planning Process	6.3.1 Project Planning (note: the mapping for this clause is only applicable for the planning of testing) 6.3.1.3.2 Project planning **6.3.1.3.2.1**	6.3.1 Project Planning 6.3.1.3.2 Project planning **6.3.1.3.2.1**	5.1.1 Software development plan
This is supported by the clause 7.2.4.2 Organize Test Plan Development (TP2), which includes the identification and notification of people who will be involved in risk management for testing-related risks.	7 Test Management Processes 7.2 Test Planning Process 7.2.4.2 Organize Test Plan Development (TP2)	6.3.4 Risk Management 6.3.4.3.1 Risk management planning **6.3.4.3.1.3** **6.3.4.3.1.4**	6.3.4 Risk Management	5.1.7 Software risk management planning
This is supported by clause 7.2.4.3 Identify and Analyse Risks (TP3), which provides a generic process for identifying and analysing testing- related risks.	7 Test Management Processes 7.2 Test Planning Process 7.2.4.3 Identify and Analyse Risks (TP3)	6.3.4 Risk Management 6.3.4.3.3 Risk analysis **6.3.4.3.3.1**	6.3.4 Risk Management	5.1.7 Software risk management planning
This is supported by clause 7.2.4.3 Identify and Analyse Risks (TP3), which allows risks to be assigned a level of exposure (e.g. impact and likelihood).	7 Test Management Processes 7.2 Test Planning Process 7.2.4.3 Identify and Analyse Risks (TP3)	6.3.4 Risk Management 6.3.4.3.3 Risk analysis **6.3.4.3.3.2**	6.3.4 Risk Management	5.1.7 Software risk management planning
The identification of testing-related risks is supported by clause 7.2.4.3 Identify and Analyse Risks (TP3), which can include the identification of risks that can be treated by operational testing.	7 Test Management Processes 7.2.4 Activities and tasks 7.2.4.5 Identify and analyse Risks (TP3)	6.4.9 Software Operation 6.4.9.3.3 Operational use **6.4.9.3.3.1**	N/A	N/A
This is supported by clause 7.2.4.3 Identify and Analyse Risks (TP3), which provides a generic process for risk identification.	7 Test Management Processes 7.2.4.5 Identify and Analyse Risks (TP3)	7.2.5 Software Validation 7.2.5.3.1 Process implementation **7.2.5.3.1.1**	N/A	N/A
This is supported by clause 7.2.4.4 Identify Risk Mitigation Approaches (TP4), which enables the identification of risk mitigation approaches for testing.	7 Test Management Processes 7.2 Test Planning Process 7.2.4.4 Identify Risk Mitigation Approaches (TP4)	6.3.4 Risk Management 6.3.4.3.3 Risk analysis **6.3.4.3.3.3** **6.3.4.3.3.4**	6.3.4 Risk Management	5.1.7 Software risk management planning

This is supported by clause 7.2.4.4 Identify Risk Mitigation Approaches (TP4), which ensures the various risk mitigation approaches that are proposed are recorded in the draft test plan.	7 Test Management Processes 7.2 Test Planning Process 7.2.4.4 Identify Risk Mitigation Approaches (TP4)	6.3.4 Risk Management 6.3.4.3.4 Risk mitigation **6.3.4.3.4.1** **6.3.4.3.4.2**	6.3.4 Risk Management	5.1.7 Software risk management planning
This is also supported by clause 7.2.4.4 Identify Risk Mitigation Approaches (TP4), which provides a generic process for risk analysis, which can be used to determine phases and types of testing that are required to treat the identified risks.	7 Test Management Processes 7.2.4.5 Identify Risk Mitigation Approaches (TP4)	7.2.5 Software Validation 7.2.5.3.1 Process implementation **7.2.5.3.1.1**	N/A	N/A
This is supported by clause 7.2.4.5 Design Test Strategy (TP5), which provides a generic process for designing a strategy for testing.	7 Test Management Processes 7.2.4.5 Design Test Strategy (TP5)	7.2.4 Software Verification 7.2.4.3.1 Process implementation **7.2.4.3.1.4**	7.2.4 Software Verification 7.2.4.3.1 Process implementation **7.2.4.3.1.4**	5.1.6 Software verification planning
This is also supported by clause 7.2.4.5 Design Test Strategy (TP5), which provides a generic process for designing a test strategy that includes a determination of the level of independence and effort required during testing.	7 Test Management Processes 7.2.4.5 Design Test Strategy (TP5)	7.2.5 Software Validation 7.2.5.3.1 Process implementation **7.2.5.3.1.1**	N/A	N/A
This is also supported by clause 7.2.4.8 Gain Consensus on Test Plan (TP8), which allows stakeholders to comment on the various risk mitigation approaches that are recorded in the draft test plan.	7 Test Management Processes 7.2 Test Planning Process 7.2.4.8 Gain Consensus on Test Plan (TP8)	6.3.4 Risk Management 6.3.4.3.4 Risk mitigation **6.3.4.3.4.1** **6.3.4.3.4.2**	6.3.4 Risk Management	5.1.7 Software risk management planning
This is also supported by clause 7.2.4.8 Gain Consensus on Test Plan (TP8), which provides an ability to gain stakeholder approval of all changes to the planned test approach, including measurement activities and resources.	7 Test Management Processes7.2 Test Planning Process7.2.4.8 Gain Consensus on Test Plan (TP8)	6.3.7 Measurement6.3.7.3.1 Measurement planning **6.3.7.3.1.6** **6.3.7.3.1.7**	N/A	N/A
This is supported by clause 7.2 Test Planning Process, which provides a generic process for planning testing, which can include the identification and documentation of testing requirements, which can cover qualification requirements.	7 Test Management Processes 7.2 Test Planning Process	6.4.2 System Requirements Analysis 6.4.2.3.1 Requirements specification **6.4.2.3.1.1**	N/A	N/A
For planning integration testing, this clause is supported by clause 7 Test Management Processes, which provides generic processes for planning testing, including integration testing.	7 Test Management Processes	6.4.5 System Integration 6.4.5.3.1 Integration **6.4.5.3.1.1**	6.4.5 System Integration 6.4.5.3.1 Integration **6.4.5.3.1.1**	5.1.3 Software development plan reference to system design and development
For planning installation testing, this clause is supported by clause 7 Test Management Processes, which provides generic processes for planning testing, including installation testing.	7 Test Management Processes	6.4.7 Software Installation 6.4.7.3.1 Software installation **6.4.7.3.1.2**	N/A	N/A
For planning testing of the software in its operational environment, this clause is supported by clause 7 Test Management Processes, which provides generic processes for planning testing, including operation testing.	7 Test Management Processes	6.4.9 Software Operation 6.4.9.3.1 Preparation and Operation **6.4.9.3.1.3**	N/A	N/A

For planning testing-related activities, this clause is supported by clause 7 Test Management Processes, which provides generic processes for planning the activities and tasks involved in maintenance testing.	7 Test Management Processes	6.4.10 Software Maintenance Process 6.4.10.3.1 Process implementation **6.4.10.3.1.1**	6.4.10 Software Maintenance Process	6.1 Establish software maintenance plan
The testing-related planning elements of this clause can be supported by clause 7.2 Test Planning Process, which can be used to prepare and communicate test plans for migration verification testing.	7 Test Management Processes 7.2 Test Planning Process	6.4.10 Software Maintenance 6.4.10.3.5 Migration **6.4.10.3.5.2** **6.4.10.3.5.3**	N/A	N/A
The testing-related planning elements of this clause can be supported by clause 7.2 Test Planning Process.	7 Test Management Processes 7.2 Test Planning Process	7.1.1 Software Implementation Process 7.1.1.3.1 Software implementation strategy **7.1.1.3.1.4**	7.1.1 Software Implementation 7.1.1.3.1 Software implementation strategy **7.1.1.3.1.4**	5.1.1 Software development plan
This is supported by clause 7.2 Test Planning Process, which provides a generic process for planning testing, which can include the identification and documentation of testing requirements, including qualification requirements.	7 Test Management Processes 7.2 Test Planning Process	7.1.2 Software Requirements Analysis 7.1.2.3.1 Software requirements analysis **7.1.2.3.1.1**	7.1.2 Software Requirements Analysis 7.1.2.3.1 Software requirements analysis **7.1.2.3.1.1**	5.2.2 Software requirements content
This is supported by clause 7.2 Test Planning Process, which provides a generic process for planning testing, which can include the identification and documentation of testing requirements, including qualification requirements.	7 Test Management Processes 7.2 Test Planning Process	7.1.2 Software Requirements Analysis 7.1.2.3.1 Software requirements analysis **7.1.2.3.1.1**	7.1.2 Software Requirements Analysis 7.1.2.3.1 Software requirements analysis **7.1.2.3.1.1**	5.2.3 Include risk control measures in software requirements
This is supported by clause 7.2 Test Planning Process, which defines a generic process for test planning, which can be used to identify test requirements and schedules for software integration testing.	7 Test Management Processes 7.2 Test Planning Process	7.1.3 Software Architectural Design 7.1.3.3.1 Software architectural design **7.1.3.3.1.5**	N/A	N/A
This is supported by clause 7.2 Test Planning Process, which defines a generic process for test planning, which can be used to identify test requirements and schedules for testing software units.	7 Test Management Processes 7.2 Test Planning Process	7.1.4 Software Detailed Design 7.1.4.3.1 Software detailed design **7.1.4.3.1.5**	7.1.4 Software Detailed Design 7.1.4.3.1 Software detailed design **7.1.4.3.1.5**	5.5.2 Establish software unit verification process
This is supported by clause 7.2 Test Planning Process, which is a generic test planning process that can be used for defining a test plan for integration testing, including test requirements for integration testing.	7 Test Management Processes 7.2 Test Planning Process	7.1.6 Software Integration 7.1.6.3.1 Software integration **7.1.6.3.1.1**	7.1.6 Software Integration 7.1.6.3.1 Software integration **7.1.6.3.1.1**	5.1.5 Software integration and integration testing planning
This is supported by clause 7.2 Test Planning Process, which is a generic test planning process that can be used for defining a test plan for quality assurance activities that are related to testing.	7 Test Management Processes 7.2 Test Planning Process	7.2.3 Software Quality Assurance 7.2.3.3.1 Process implementation **7.2.3.3.1.3**	N/A	N/A
The initiation, management and control of any type of testing can be supported by clause 7.2 Test Monitoring and Control Process.	7 Test Management Processes 7.2 Test Planning Process	7.2.3 Software Quality Assurance 7.2.3.3.1 Process implementation **7.2.3.3.1.4**	N/A	N/A

Ensuring that records of any type of testing are made available to the acquirer can be supported by clause 7.2 Test Monitoring and Control Process.	7 Test Management Processes 7.2 Test Planning Process	7.2.3 Software Quality Assurance 7.2.3.3.1 Process implementation **7.2.3.3.1.5**	N/A	N/A
Ensuring that testing staff have freedom, resources and authority to permit the required testing can be supported by clause 7.2 Test Monitoring and Control Process.	7 Test Management Processes 7.2 Test Planning Process	7.2.3 Software Quality Assurance 7.2.3.3.1 Process implementation **7.2.3.3.1.6**	N/A	N/A
Ensuring that test plans are executed as required can be supported by clause 7.2 Test Monitoring and Control Process.	7 Test Management Processes 7.2 Test Planning Process	7.2.3 Software Quality Assurance 7.2.3.3.2 Product assurance **7.2.3.3.2.1**	N/A	N/A
This is supported by clause 7.2 Test Planning Process, which supports the identification of testing-related risks, consideration of the available testing budget and resources, and the selection of appropriate teams for carrying out the testing.	7 Test Management Processes 7.2 Test Planning Process	7.2.4 Software Verification 7.2.4.3.1 Process implementation **7.2.4.3.1.1** **7.2.4.3.1.2** **7.2.4.3.1.3**	N/A	N/A
This is supported by clause 7.2 Test Planning Process, which can include steps for deciding on how to staff the required testing, particularly clause 7.2.4.6 Determine Staffing and Scheduling (TP6), to support the selection of appropriate staff for verification testing.	7 Test Management Processes 7.2 Test Planning Process	7.2.4 Software Verification 7.2.4.3.1 Process implementation **7.2.4.3.1.3**	N/A	N/A
This is supported by clause 7.2 Test Planning Process, which includes generic activities for designing test plans, which can be used to design a test plan to suit the required verification effort.	7 Test Management Processes 7.2 Test Planning Process	7.2.4 Software Verification 7.2.4.3.1 Process implementation **7.2.4.3.1.5**	7.2.4 Software Verification 7.2.4.3.1 Process implementation **7.2.4.3.1.5**	5.1.6 Software verification planning
The initiation, management and control of verification testing can be supported by clause 7.2 Test Monitoring and Control Process.	7 Test Management Processes 7.2 Test Monitoring and Control Process	7.2.4 Software Verification 7.2.4.3.1 Process implementation **7.2.4.3.1.6**	7.2.4 Software Verification 7.2.4.3.1 Process implementation **7.2.4.3.1.6**	5.6.8 Use software problem resolution process
The initiation, management and control of verification testing can be supported by clause 7.2 Test Monitoring and Control Process.	7 Test Management Processes 7.2 Test Monitoring and Control Process	7.2.4 Software Verification 7.2.4.3.1 Process implementation **7.2.4.3.1.6**	7.2.4 Software Verification 7.2.4.3.1 Process implementation **7.2.4.3.1.6**	5.7.2 Use software problem resolution process
This is supported by clause 7.2 Test Planning Process, which includes generic activities for designing test plans, which can be used to design a test plan to suit the required validation effort.	7 Test Management Processes 7.2 Test Planning Process	7.2.5 Software Validation 7.2.5.3.1 Process implementation **7.2.5.3.1.2**	N/A	N/A
This is supported by clause 7.2 Test Planning Process, which can include steps for deciding on how to staff the required testing, particularly clause 7.2.4.6 Determine Staffing and Scheduling (TP6), to support the selection of appropriate staff for validation testing.	7 Test Management Processes 7.2 Test Planning Process	7.2.5 Software Validation 7.2.5.3.1 Process implementation **7.2.5.3.1.3**	N/A	N/A
This is supported by clause 7.2 Test Planning Process, which includes generic activities for designing test plans, which can be used to design a test plan to suit the required validation effort.	7 Test Management Processes7.2 Test Planning Process	7.2.5 Software Validation 7.2.5.3.1 Process implementation **7.2.5.3.1.4**	7.2.5 Software Validation 7.2.5.3.1 Process implementation **7.2.5.3.1.4**	5.1.3 Software development plan reference to system design and development

This is also supported by clause 7.2 Test Planning Process, which defines a generic process for test planning, which can be used to identify test requirements for validation testing, prior to the commencement of test case design.	7 Test Management Processes 7.2 Test Planning Process	7.2.5 Software Validation 7.2.5.3.2 Validation **7.2.5.3.2.1**	N/A	N/A
This is supported by clause 7.3 Test Monitoring and Control Process, which supports monitoring testing progress and defect management process.	7 Test Management Processes 7.3 Test Monitoring and Control Process	6.1.2 Supply 6.1.2.3.4 Contract execution **6.1.2.3.4.8**	N/A	N/A
The initiation, management and control of validation testing can be supported by clause 7.3 Test Monitoring and Control Process.	7 Test Management Processes 7.3 Test Monitoring and Control Process	7.2.5 Software Validation 7.2.5.3.1 Process implementation **7.2.5.3.1.5**	N/A	N/A
This is supported by clause 7.3 Test Monitoring and Control Process, which supports monitoring testing progress throughout the project.	7 Test Management Processes 7.3 Test Monitoring and Control Process	7.2.6 Software Review 7.2.6.3.2 Project Management Reviews **7.2.6.3.2.1**	N/A	N/A
This is supported by clause 7.3.4.1 Set-Up (TMC1), which ensures suitable measures for monitoring risk mitigation are identified.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.1 Set-Up (TMC1)	6.3.4 Risk Management 6.3.4.3.5 Risk monitoring **6.3.4.3.5.2**	6.3.4 Risk Management	5.1.7 Software risk management planning
This is supported by clause 7.3.4.1 Set-Up (TMC1), which enables the planning of measurement collection for testing.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.1 Set-Up (TMC1)	6.3.7 Measurement 6.3.7.3.1 Measurement planning **6.3.7.3.1.1** **6.3.7.3.1.2** **6.3.7.3.1.3** **6.3.7.3.1.4** **6.3.7.3.1.5**	N/A	N/A
This is also supported by clause 7.3.4.1 Set-Up (TMC1), which provides the ability to establish the collection of measures during testing.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.1 Set-Up (TMC1)	6.3.7 Measurement 6.3.7.3.1 Measurement planning **6.3.7.3.1.6** **6.3.7.3.1.7**	N/A	N/A
The monitoring of testing is supported by clause 7.3.4.2 Monitor (TMC2), which provides generic tasks for monitoring progress within any phase or type of testing.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.1 Set-Up (TMC1)	6.3.2 Project Assessment and Control 6.3.2.3.2 Project monitoring **6.3.2.3.1.1**	N/A	N/A
This is supported by clause 7.3.4.2 Monitor (TMC2), which ensures mitigation of testing- related risks are carried out and monitored and ensures that new testing-related risks that are identified are recorded throughout testing.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.2 Monitor (TMC2)	6.3.4 Risk Management 6.3.4.3.4 Risk mitigation **6.3.4.3.4.3** **6.3.4.3.4.4** 6.3.4.3.5 Risk monitoring **6.3.4.3.5.1** **6.3.4.3.5.3**	6.3.4 Risk Management	5.1.7 Software risk management planning

This is supported by clause 7.3.4.2 Monitor (TMC2), which provides a generic process for determining aspects such as whether testing is complete and whether the testing process has been appropriate.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.2 Monitor (TMC2)	6.4.5 System Integration 6.4.5.3.2 Test readiness **6.4.5.3.2.2**	N/A	N/A
This is supported by clause 7.3.4.2 Monitor (TMC2), which provides a generic process for monitoring testing, which can be used to determine test coverage and determine whether expected and actual results of testing match.	7 Test Management Processes7.3 Test Monitoring and Control Process7.3.4.2 Monitor (TMC2)	6.4.6 System Qualification Testing 6.4.6.3.1 Qualification testing **6.4.6.3.1.2**	N/A	N/A
This is supported by clause 7.3.4.2 Monitor (TMC2), which could be used to determine whether a developer is supporting an acquirer's acceptance review and testing.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.2 Monitor (TMC2)	6.4.8 Software Acceptance Support 6.4.8.3.1 Software acceptance support **6.4.8.3.1.1**	N/A	N/A
The monitoring of testing-related risks during operational testing is supported by clause 7.3.4.2 Monitor (TMC2), which provides generic tasks for monitoring progress within any phase or type of testing.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.2 Monitor (TMC2)	6.4.9 Software Operation 6.4.9.3.3 Operational use **6.4.9.3.3.1**	N/A	N/A
This can be supported by clause 7.3.4.2 Monitor (TMC2), which could be used to determine whether an implementer is conducting reviews according to a specific process.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.2 Monitor (TMC2)	7.1.2 Software Requirements Analysis 7.1.2.3.1 Software requirements analysis **7.1.2.3.1.3**	N/A	N/A
This can be supported by clause 7.3.4.2 Monitor (TMC2), which could be used to determine whether an implementer is conducting reviews according to a specific process.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.2 Monitor (TMC2)	7.1.3 Software Architectural Design 7.1.3.3.1 Software architectural design **7.1.3.3.1.7**	N/A	N/A
The monitoring of testing is supported by clause 7.3.4.2 Monitor (TMC2), which provides generic tasks for monitoring progress within any phase or type of testing, which can include monitoring the level of test coverage achieved.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.2 Monitor (TMC2)	7.1.5 Software Construction 7.1.5.3.1 Software construction **7.1.5.3.1.5**	7.1.5 Software Construction 7.1.5.3.1 Software construction **7.1.5.3.1.5**	5.1.6 Software verification planning
The monitoring of testing is supported by clause 7.3.4.2 Monitor (TMC2), which provides generic tasks for monitoring progress within any phase or type of testing, which can include monitoring the level of test coverage achieved.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.2 Monitor (TMC2)	7.1.5 Software Construction 7.1.5.3.1 Software construction **7.1.5.3.1.5**	7.1.5 Software Construction 7.1.5.3.1 Software Construction **7.1.5.3.1.5**	5.5.2 Establish software unit verification process
The monitoring of testing is supported by clause 7.3.4.2 Monitor (TMC2), which provides generic tasks for monitoring progress within any phase or type of testing, which can include monitoring the level of test coverage achieved.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.2 Monitor (TMC2)	7.1.5 Software Construction 7.1.5.3.1 Software construction **7.1.5.3.1.5**	7.1.5 Software Construction 7.1.5.3.1 Software construction **7.1.5.3.1.5**	5.5.3 software unit acceptance criteria
The monitoring of testing is supported by clause 7.3.4.2 Monitor (TMC2), which provides generic tasks for monitoring progress within any phase or type of testing, which can include monitoring the level of test coverage achieved.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.2 Monitor (TMC2)	7.1.6 Software Integration 7.1.6.3.1 Software integration **7.1.6.3.1.5**	7.1.6 Software Integration 7.1.5.3.1 Software construction **7.1.5.3.1.5**	5.1.6 Software verification planning

The management control of testing is supported by clause 7.3.4.3 Control (TMC3), which provides generic tasks for controlling any phase or type of testing.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.3 Control (TMC3)	6.3.2 Project Assessment and Control 6.3.2.3.2 Project control **6.3.2.3.2.1**	6.3.2 Project Assessment and Control 6.3.2.3.2 Project control **6.3.2.3.2.1**	5.1.2 Keep software development plan updated
This is supported by clause 7.3.4.3 Control (TMC3), which provides a generic process for making changes to the test approach, which can include updating test plans with required changes to schedules.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.3 Control (TMC3)	7.1.4 Software Detailed Design 7.1.4.3.1 Software detailed design **7.1.4.3.1.6**	N/A	N/A
This is supported by clause 7.3.4.3 Control (TMC3), which provides a generic process for making changes to the test approach, which can include updating test plans with required changes to schedules.	7 Test Management Processes7.3 Test Monitoring and Control Process7.3.4.3 Control (TMC3)	7.1.5 Software Construction 7.1.5.3.1 Software construction **7.1.5.3.1.4**	N/A	N/A
This is supported by clause 7.3.4.4 Report (TMC4), which provides an ability to report testing progress and communicate new risks to stakeholders.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.4 Report (TMC4)	6.1.2 Supply 6.1.2.3.4 Contract execution **6.1.2.3.4.15**	N/A	N/A
The reporting of progress during testing is supported by clause 7.3.4.4 Report (TMC4), which provides generic tasks for reporting on the progress of any phase or type of testing.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.4 Report (TMC4)	6.3.2 Project Assessment and Control 6.3.2.3.2 Project control **6.3.2.3.2.2**	N/A	N/A
7.3.4.4 Report (TMC4) This activity consists of the following tasks: … g) b) New risks and changes to existing risks shall be updated in the risk registry and communicated to the relevant stakeholders.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.4 Report (TMC4)	6.3.4 Risk Management 6.3.4.3.2 Risk profile management **6.3.4.3.2.4**	6.3.4 Risk Management	5.1.7 Software risk management planning
This is supported by clause 7.3.4.4 Report (TMC4), which enables the communication of measurement results in testing to relevant stakeholders.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.4 Report (TMC4)	6.3.7 Measurement 6.3.7.3.2 Measurement performance **6.3.7.3.2.4**	N/A	N/A
This can be supported by clause 7.3.4.4 Report (TMC4), which can be used to report on analysis of defect trends throughout testing.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.4 Report (TMC4)	7.2.8 Software Problem Resolution 7.2.8.3.1 Process implementation **7.2.8.3.1.1**	7.2.8 Software Problem Resolution 7.2.8.3.1 Process implementation **7.2.8.3.1.1**	5.1.9 Software configuration management planning
This can be supported by clause 7.3.4.4 Report (TMC4), which can be used to report on analysis of defect trends throughout testing.	7 Test Management Processes 7.3 Test Monitoring and Control Process 7.3.4.4 Report (TMC4)	7.2.8 Software Problem Resolution 7.2.8.3.1 Process implementation **7.2.8.3.1.1**	7.2.8 Software Problem Resolution 7.2.8.3.1 Process implementation **7.2.8.3.1.1**	5.7.3 Retest after changes
This is supported by clause 7.4.4.3 Identify Lessons Learned (TC3), which enables the identification of approaches to improve the risk management process for testing.	7 Test Management Processes 7.4 Test Completion Process 7.4.4.3 Identify Lessons Learned (TC3)	6.3.4 Risk Management 6.3.4.3.6 Risk management process evaluation **6.3.4.3.6.1**	6.3.4 Risk Management	5.1.7 Software risk management planning

This is supported by clause 7.4.4.3 Identify Lessons Learned (TC3), which allows organizational risks to be identified and communicated to the persons responsible for maintaining the organizational test specifications (e.g. to document testing-related risks that are applicable to the organization as a whole to be recorded in the organizational test strategy).	7 Test Management Processes 7.4 Test Completion Process 7.4.4.3 Identify Lessons Learned (TC3)	6.3.4 Risk Management 6.3.4.3.6 Risk management process evaluation **6.3.4.3.6.3**	6.3.4 Risk Management	5.1.7 Software risk management planning
This is supported by clause 7.4.4.3 Identify Lessons Learned (TC3), which facilitates the analysis and recording of improvements to measurement activities in testing, and to communicate them to relevant stakeholders.	7 Test Management Processes 7.4 Test Completion Process 7.4.4.3 Identify Lessons Learned (TC3)	6.3.7 Measurement 6.3.7.3.3 Measurement evaluation **6.3.7.3.3.1** **6.3.7.3.3.2**	N/A	N/A
This is also supported by clause 7.4.4.4 Report Test Completion (TC4), which provides the ability to report the outcomes of testing to stakeholders.	7 Test Management Processes 7.4 Test Completion Process 7.4.4.4 Report Test Completion (TC4)	6.1.2 Supply 6.1.2.3.4 Contract execution **6.1.2.3.4.15**	N/A	N/A
This is also supported by clause 7.4.4.4 Report Test Completion (TC4), which provides a generic process for reporting on the results of testing to required stakeholders.	7 Test Management Processes7.4 Test Completion Process7.4.4.4 Report Test Completion (TC4)	7.2.5 Software Validation 7.2.5.3.1 Process implementation **7.2.5.3.1.5**	N/A	N/A
This can be supported by clause 7.4.4.4 Report Test Completion (TC4), which can be used to report on analysis of defect trends at the end of a testing phase or project.	7 Test Management Processes 7.4 Test Completion Process 7.4.4.4 Report Test Completion (TC4)	7.2.8 Software Problem Resolution 7.2.8.3.1 Process implementation **7.2.8.3.1.1**	7.2.8 Software Problem Resolution 7.2.8.3.1 Process implementation **7.2.8.3.1.1**	5.1.9 Software configuration management planning
This can be supported by clause 7.4.4.4 Report Test Completion (TC4), which can be used to report on analysis of defect trends at the end of a testing phase or project.	7 Test Management Processes 7.4 Test Completion Process 7.4.4.4 Report Test Completion (TC4)	7.2.8 Software Problem Resolution 7.2.8.3.1 Process implementation **7.2.8.3.1.1**	7.2.8 Software Problem Resolution 7.2.8.3.1 Process implementation **7.2.8.3.1.1**	5.7.3 Retest after changes
This is supported by clause 8 Dynamic Test Processes, which provides a generic process for testing, which can be used for integration testing.	8 Dynamic Test Processes	6.4.5 System Integration 6.4.5.3.1 Integration **6.4.5.3.1.1**	6.4.5 System Integration 6.4.5.3.1 Integration **6.4.5.3.1.1**	5.1.3 Software development plan reference to system design and development
This is supported by clause 8.5 Test Incident Reporting Process, which provides a generic process for reporting incidents that are detected throughout testing.	8 Dynamic Test Processes 8.5 Test Incident Reporting Process	7.2.8 Software Problem Resolution 7.2.8.3.1 Process implementation **7.2.8.3.1.1**	7.2.8 Software Problem Resolution 7.2.8.3.1 Process implementation **7.2.8.3.1.1**	5.1.9 Software configuration management planning
This is supported by clause 8.5 Test Incident Reporting Process, which provides a generic process for reporting incidents, which can be used for reporting and categorising problems.	8 Dynamic Test Processes 8.5 Test Incident Reporting Process 8.5.4 Activities and tasks	7.2.8 Software Problem Resolution 7.2.8.3.1 Process implementation **7.2.8.3.1.1**	7.2.8 Software Problem Resolution 7.2.8.3.1 Process implementation **7.2.8.3.1.1**	5.1.9 Software configuration management planning
This is supported by clause 8 Dynamic Test Processes, which includes generic processes for test design and implementation, and which can be used for designing test procedures and data for each software unit and database.	8 Dynamic Test Processes	7.1.5 Software Construction 7.1.5.3.1 Software construction **7.1.5.3.1.1**	7.1.5 Software Construction 7.1.5.3.1 Software construction **7.1.5.3.1.1**	5.5.1 Implement each software unit
This is supported by clause 8 Dynamic Test Processes, which provide generic processes that can be used for any phase or type of testing.	8 Dynamic Test Processes	7.1.5 Software Construction 7.1.5.3.1 Software construction **7.1.5.3.1.2**	7.1.5 Software Construction 7.1.5.3.1 Software construction **7.1.5.3.1.2**	5.5.4 Additional software unit acceptance criteria

This is supported by clause 8 Dynamic Test Processes, which provide generic processes that can be used for any phase or type of testing.	8 Dynamic Test Processes	7.1.5 Software Construction 7.1.5.3.1 Software construction **7.1.5.3.1.2**	7.1.5 Software Construction 7.1.5.3.1 Software construction **7.1.5.3.1.2**	5.5.5 Software unit verification
This is supported by clause 8 Dynamic Test Processes, which includes generic processes for the design, documentation and execution of test cases, including integration test cases.	8 Dynamic Test Processes	7.1.6 Software Integration 7.1.6.3.1 Software integration **7.1.6.3.1.2**	7.1.6 Software Integration 7.1.6.3.1 Software integration **7.1.6.3.1.2**	5.6.1 Integrate software units
This is supported by clause 8 Dynamic Test Processes, which includes generic processes for the design, documentation and execution of test cases, including integration test cases.	8 Dynamic Test Processes	7.1.6 Software Integration 7.1.6.3.1 Software integration **7.1.6.3.1.2**	7.1.6 Software Integration 7.1.6.3.1 Software integration **7.1.6.3.1.2**	5.6.2 Verify software integration
This is supported by clause 8 Dynamic Test Processes, which provides a generic process for testing, which can be used for integration testing.	8 Dynamic Test Processes	6.4.5 System Integration 6.4.5.3.1 Integration **6.4.5.3.1.1**	6.4.5 System Integration 6.4.5.3.1 Integration **6.4.5.3.1.1**	5.6.2 Verify software integration
This is also supported by clause 8 Dynamic Test Processes, which defines generic processes for carrying out test design, implementation and execution, which includes software qualification testing.	8 Dynamic Test Processes	7.1.7 Software Qualification Testing 7.1.7.3.1 Software qualification testing **7.1.7.3.1.1**	7.1.7 Software Qualification Testing 7.1.7.3.1 Software qualification testing **7.1.7.3.1.1**	5.6.3 Test integrated software
This is supported by clause 8 Dynamic Test Processes, which includes generic processes for the design, documentation and execution of test cases, including integration test cases.	8 Dynamic Test Processes	7.1.6 Software Integration 7.1.6.3.1 Software integration **7.1.6.3.1.2**	7.1.6 Software Integration 7.1.6.3.1 Software integration **7.1.6.3.1.2**	5.6.6 Conduct regression tests
This is supported by clause 8 Dynamic Test Processes, which includes generic processes for the design, documentation and execution of test cases, including integration test cases.	8 Dynamic Test Processes	7.1.6 Software Integration 7.1.6.3.1 Software integration **7.1.6.3.1.2**	7.1.6 Software Integration 7.1.6.3.1 Software integration **7.1.6.3.1.2**	5.6.7 Integration test record contents
This is supported by clause 8 Dynamic Test Processes, which includes generic processes for the design and documentation of test cases, including qualification test cases and test procedures.	8 Dynamic Test Processes	7.2.4 Software Verification 7.2.4.3.1 Process implementation **7.2.4.3.1.6**	7.2.4 Software Verification 7.2.4.3.1 Process implementation **7.2.4.3.1.6**	5.6.8 Use software problem resolution process
This is supported by clause 8 Dynamic Test Processes, which includes generic processes for the design and documentation of test cases, including qualification test cases and test procedures.	8 Dynamic Test Processes	7.1.6 Software Integration 7.1.6.3.1 Software integration **7.1.6.3.1.4**	7.1.6 Software Integration 7.1.6.3.1 Software integration **7.1.6.3.1.4**	5.7.1 Establish tests for each software requirement
This is also supported by clause 8 Dynamic Test Processes, which defines generic processes for carrying out test design, implementation and execution, which includes software qualification testing.	8 Dynamic Test Processes	7.1.7 Software Qualification Testing 7.1.7.3.1 Software qualification testing **7.1.7.3.1.1**	7.1.7 Software Qualification Testing 7.1.7.3.1 Software qualification testing **7.1.7.3.1.1**	5.7.1 Establish tests for each software requirement
This is supported by clause 8 Dynamic Test Processes, which includes generic processes for the design and documentation of test cases, including qualification test cases and test procedures.	8 Dynamic Test Processes	7.2.4 Software Verification 7.2.4.3.1 Process implementation **7.2.4.3.1.6**	7.2.4 Software Verification 7.2.4.3.1 Process implementation **7.2.4.3.1.6**	5.7.2 Use software problem resolution process
This is supported by clause 8.5 Test Incident Reporting Process, which provides a generic process for reporting incidents that are detected throughout testing.	8 Dynamic Test Processes 8.5 Test Incident Reporting Process	7.2.8 Software Problem Resolution 7.2.8.3.1 Process implementation **7.2.8.3.1.1**	7.2.8 Software Problem Resolution 7.2.8.3.1 Process implementation **7.2.8.3.1.1**	5.7.3 Retest after changes
This is supported by clause 8.5 Test Incident Reporting Process, which provides a generic process for reporting incidents, which can be used for reporting and categorising problems.	8 Dynamic Test Processes 8.5 Test Incident Reporting Process	7.2.8 Software Problem Resolution 7.2.8.3.1 Process implementation **7.2.8.3.1.1**	7.2.8 Software Problem Resolution 7.2.8.3.1 Process implementation **7.2.8.3.1.1**	5.7.3 Retest after changes

This is also supported by clause 8 Dynamic Test Processes, which defines generic processes for carrying out test design, implementation and execution, which includes software qualification testing.	8 Dynamic Test Processes	7.1.7 Software Qualification Testing 7.1.7.3.1 Software qualification testing **7.1.7.3.1.1**	7.1.7 Software Qualification Testing 7.1.7.3.1 Software qualification testing **7.1.7.3.1.1**	5.7.5 software system test record contents
This is supported by clause 8 Dynamic Test Processes, which could be used for testing the software in the operation environment.	8 Dynamic Test Processes	6.4.9 Software Operation 6.4.9.3.2 Operation activation and check-out **6.4.9.3.2.1** **6.4.9.3.2.2**	6.4.9 Software Operation 6.4.9.3.2 Operation activation and check-out **6.4.9.3.2.1**	5.8.1 Ensure software verification is complete
This is supported by clause 8 Dynamic Test Processes, which could be used for testing the software in the operation environment.	8 Dynamic Test Processes	6.4.9 Software Operation 6.4.9.3.2 Operation activation and check-out **6.4.9.3.2.1** **6.4.9.3.2.2**	6.4.9 Software Operation 6.4.9.3.2 Operation activation and check-out **6.4.9.3.2.2**	5.8.1 Ensure software verification is complete
This is supported by clause 8 Dynamic Test Processes, which defines a generic test design, implementation and execution process that can be used to demonstrate that changes and fixes as a result of maintenance have not compromised the ability of the software to meet its requirements.	8 Dynamic Test Processes	6.4.10 Software Maintenance 6.4.10.3.2 Problem and modification analysis	6.4.10 Software Maintenance	6.2.1.1 Monitor feedback
	8 Dynamic Test Processes	6.4.10 Software Maintenance 6.4.10.3.2 Problem and modification analysis	6.4.10 Software Maintenance	6.2.1.2 Document and evaluate feedback
	8 Dynamic Test Processes	6.4.10 Software Maintenance 6.4.10.3.2 Problem and modification analysis	6.4.10 Software Maintenance	6.2.1.3 Evaluate problem report's effects on safety
	8 Dynamic Test Processes	6.4.10 Software Maintenance 6.4.10.3.2 Problem and modification analysis	6.4.10 Software Maintenance	6.2.2 Use software problem resolution process
	8 Dynamic Test Processes	6.4.10 Software Maintenance 6.4.10.3.2 Problem and modification analysis	6.4.10 Software Maintenance	6.2.3 Analyse change requests
	8 Dynamic Test Processes	6.4.10 Software Maintenance6.4.10.3.2 Problem and modification analysis	6.4.10 Software Maintenance	6.2.4 Change request approval
	8 Dynamic Test Processes	6.4.10 Software Maintenance 6.4.10.3.2 Problem and modification analysis	6.4.10 Software Maintenance	6.2.5 Communicate to users and regulators
	8 Dynamic Test Processes	6.4.10 Software Maintenance 6.4.10.3.2 Problem and modification analysis	6.4.10 Software Maintenance	6.3.1 Use established process to implement modification

Appendix B: MED-V-STEP in Excel File

The attached CD contains the Excel file with the entire framework for the MED-V-STEP project.

Appendix C: Data Obtained from Questionnaires

How efficient do you think the MED-V-STEP framework is in providing information on needed processes, activities and tasks and their relationship? Range 1-5	
***Questionnaire 1** with Medical Device Software Development Organisation*	
Participant 1	4
Participant 2	4
Participant 3	4
Participant 4	3
Participant 5	4
Participant 6	4
***Questionnaire 2** with Software Testing Organisation*	
Participant 1	5
Participant 2	5
How efficient do you think the MED-V-STEP framework is in improving industry software testing practices? Range 1-5	
***Questionnaire 1** with Medical Device Software Development Organisation*	
Participant 1	3
Participant 2	5
Participant 3	4
Participant 4	2
Participant 5	5
Participant 6	4
***Questionnaire 2** with Software Testing Organisation*	
Participant 1	5
Participant 2	5
How efficient do you think the MED-V-STEP framework is in identification of how software test activities of ISO/IEC/IEEE 29119-2 need to be enhanced with requirements of IEC 62304? Range 1-5	
***Questionnaire 1** with Medical Device Software Development Organisation*	
Participant 1	4
Participant 2	4
Participant 3	4
Participant 4	3
Participant 5	4
Participant 6	3
***Questionnaire 2** with Software Testing Organisation*	
Participant 1	5
Participant 2	5